Universidad Carlos III de Madrid

Departamento de Informática

Trabajo Fin de Grado

CollectionFS

Sistema de Ficheros basado en Etiquetas

Autor: Analía de Pedro Santamaría

Tutor: Luis Miguel Sánchez García

Co-Tutor: María-Cristina Marinescu

Agradecimientos

En primer lugar quiero agradecer a mi tutor, Luis Miguel, el guiarme a lo largo de todo este proyecto, ayudándome en todo lo que he necesitado y dedicándome gran parte de su tiempo.

Así mismo, no puedo dejar de mencionar a mi familia por apoyarme en todo momento, en especial a mi padre, gracias Papá por tus consejos.

Y por último, pero no por ello menos importante, a Nico, por ayudarme a llevar el peso y estar siempre ahí, gracias.

Resumen

CollectionFS es un Sistema de Ficheros basado en Etiquetas cuyo objetivo es la organización y localización de la información de forma rápida y sencilla.

Actualmente el espacio de almacenamiento está poco limitado, lo que provoca que cada vez almacenemos más información. Cuando toda esa información almacenada tiene características muy heterogéneas como distinta extensión, tamaño, fecha de creación o diferente temática, el clasificarla para su recuperación es realmente complejo.

Los Sistemas de Ficheros tradicionales permiten una estructura jerárquica que proporciona cierta organización, pero cuando la cantidad de información es muy grande puede no ser suficiente. Por este motivo han surgido los Sistemas de Ficheros Semánticos, que clasifican la información a través de su contenido, ofreciendo un método para recuperar la información conociendo *qué* se busca en lugar de *dónde* es necesario buscar.

El sistema aquí presentado se puede clasificar como Sistema de Ficheros Semántico ya que clasifica los archivos a través de etiquetas, que definen propiedades del documento como el tamaño, la extensión o la temática. CollectionFS proporciona, por lo tanto, un sistema de almacenamiento sencillo – simplemente hay que insertar archivos, crear etiquetas y etiquetar archivos – y una forma de recuperar la información rápida y efectiva – siendo posible la búsqueda simultánea a través de diferentes propiedades y etiquetas.

Palabras clave: CollectionFS, sistema de ficheros semántico, etiquetas.

Abstract

CollectionFS is a based tag File System which aim is the quickly and simple management and localization of the information.

Currently the storage space is almost unlimited. It promotes us to store more and more information. When all of the information has heterogeneous characteristics as different extension, size, creation date or diverse topics, it is really complex to classify it.

The traditional file systems supply a hierarchical structure which provides some organization. However when the amount of information is outsize may not be sufficient. For this reason the semantic file system has appeared. This type of file system classifies the information through its content, providing a method to recover the information knowing *what* we are looking for instead of *where* we have to search.

The system presented here can be classified as a semantic file system since it classifies the files through tags, which define document properties like the size, extension or theme. Therefore CollectionFS provides a simple storage system – you just have to insert files, create tags and label files - and a simple and quickly way to recover the information – being possible the simultaneous search of different properties and tags.

Key words: CollectionFS, semantic file system, tags.

Índice

Índice de Figuras

Índice de Tablas

1. Introducción

Este documento pretende describir el proyecto “CollectionFS: Sistema de Ficheros Basado en Etiquetas”, consistente en el análisis y diseño de una herramienta capaz de gestionar y localizar documentos de forma rápida y sencilla utilizando etiquetas.

Antes, en esta presentación, se analiza la motivación que ha impulsado este trabajo, se declaran los objetivos que persigue el proyecto y se relacionan los acrónimos y términos utilizados en este documento.

1.1 Motivación

En la actualidad, debido a la gran capacidad de almacenamiento que ofrecen los equipos y sistemas informáticos, tanto personales como profesionales, se tiende de forma natural a guardar en ellos una enorme cantidad de información. Lo normal es que el número de documentos o ficheros que se almacenan vaya creciendo con el tiempo, y que las características de toda esta información en cuanto a tipo, contenido, temática o utilidad se vayan haciendo más complejas, dificultando la tarea de organizar el archivo.

El sistema tradicional de directorios, ordenados jerárquicamente, da facilidades para el manejo de la información, pero tienen limitaciones:

- La jerarquía impuesta por este sistema obliga a clasificar la información de forma rígida, ubicando cada documento en una única posición que se alcanza a través de una secuencia de directorios y subdirectorios.
- El espacio de nombres establecido en el sistema de ficheros también impone restricciones: cada archivo se identifica mediante un único nombre, lo que obliga a establecer criterios de nombrado muy precisos que faciliten la identificación.

Cuando la información almacenada en un sistema informático es abundante, heterogénea y, sobre todo, susceptible de ser clasificada mediante varios criterios simultáneamente, estas limitaciones presentan los siguientes problemas para los usuarios:

- Deben recordar dónde guardaron determinado archivo, qué nombre le pusieron, qué criterio utilizaron para nombrarlo, etc. La búsqueda y recuperación de documentos se hace, en consecuencia, más compleja y requiere más tiempo del que sería razonable.

- Al dificultarse la búsqueda es habitual que un mismo documento sea almacenado varias veces en momentos diferentes y en ubicaciones distintas, creando réplicas innecesarias y desaprovechando espacio de almacenamiento útil.
- El usuario de un sistema de ficheros convencional se enfrenta a la creación de un método de nombrado adecuado para recordar *dónde* buscar, mientras que lo natural es conocer *qué* buscar.
- Los problemas de búsqueda tienen como consecuencia habitual que los usuarios olviden de qué información disponen.

Los problemas aquí planteados crean la necesidad de un sistema de ficheros capaz de organizar la información de forma que su recuperación resulte sencilla y rápida. Esta necesidad se hace cada vez mayor [1], lo que nos conduce al desarrollo de los sistemas de ficheros semánticos, que permiten organizar y recuperar la información por su contenido [2, 3].

El funcionamiento básico de este tipo de sistemas de ficheros se apoya en dos ideas fundamentales: el almacenamiento físico en el sistema de ficheros tradicional y la clasificación semántica a través del contenido o los metadatos de los ficheros (almacenada en una Base de Datos SQL).

El presente trabajo se centrará en los sistemas de ficheros semánticos, añadiendo como novedad la utilización de una Base de Datos *NoSQL* para la gestión de los metadatos.

1.2 Objetivos

El objetivo principal de este Trabajo Fin de Grado es el análisis y diseño de un sistema de ficheros semánticos, que facilite la organización del espacio de nombres y la localización de los archivos almacenados en el sistema, permitiendo al usuario el acceso transparente al sistema de ficheros.

Para conseguir este objetivo general se ha dividido el mismo en los siguientes sub-objetivos:

- Habilitar un espacio de nombres para los datos utilizando etiquetas definidas por el usuario.
- Implementar la definición de etiquetas por defecto para los datos almacenados en el sistema, tales como extensión del fichero o clasificación en base al tamaño.
- Realizar un análisis de las tecnologías para gestionar los metadatos del sistema de ficheros, tales como etiquetas, nombre del fichero, etc., que permitan la localización

rápida de los datos Para ello se analizarán sistemas de bases de datos *SQL* (*MySQL*) y *NoSQL* (*MongoDB*).

- Estudio de integración del sistema de ficheros en el sistema operativo Linux mediante el uso de FUSE (*Filesystem in Userspace*), para el acceso transparente a los datos.
- Desarrollo ágil de un prototipo inicial para evaluar el sistema de ficheros. Para ello se analizarán los requisitos de usuario así como las restricciones del sistema.

1.3 Acrónimos

Este apartado contiene los acrónimos utilizados en el presente documento (ordenados alfabéticamente) y su significado.

Acrónimo	Significado
BD	Base de Datos
CPU	Central Processing Unit (Unidad de Procesamiento Central)
E/S	Entrada/Salida
ext	Extended File System
FAT	File Allocation Table
HPFS	High Performance File System
HTTP	Hypertext Transfer Protocol
MFT	Master File Table
NTFS	New Technology File System
PiB	Pebibyte
RDF	Resource Description Framework
RS	Requisito de Software
RU	Requisito de Usuario
SGBD	Sistema de Gestión de Bases de Datos
SGBDD	Sistema de Gestión de Bases de Datos Documentales
TFG	Trabajo Fin de Grado
SF	Sistema de Ficheros
SQL	Structured Query Language (Lenguaje de consulta estructurado)
UFS	Unix File System
VFS	Virtual File System
WebDAV	Web Distributed Authoring and Versioning

Tabla 1.1: Acrónimos

1.4 Glosario

Esta sección contiene la definición de términos utilizados en el presente documento.

Cluster: Conjunto o conglomerado de computadoras construido mediante la utilización de hardwares comunes y se comporta como una única máquina.

Diccionario: Colección que relaciona una clave y un valor.

Disco Floppy: Más conocido como 'disquete', es un medio o soporte de almacenamiento de datos formado por una pieza circular de material magnético encerrada en una cubierta de plástico cuadrada o rectangular.

Extent: Conjunto de bloques físicos contiguos, que mejora el rendimiento al trabajar con ficheros de gran tamaño y reduce la fragmentación.

I-nodo: Estructura de datos propia de los sistemas de archivos tradicionalmente empleados en los sistemas operativos tipo UNIX. Contiene las características de un archivo regular, directorio, o cualquier otro objeto que pueda contener el sistema de ficheros.

Journaling: Mecanismo por el cual un sistema informático puede implementar transacciones. Se basa en llevar un *journal* o registro de diario en el que se almacena la información necesaria para restablecer los datos afectados por la transacción en caso de que ésta falle. Una de las aplicaciones más frecuentes de los sistemas de *journaling* es evitar la corrupción de las estructuras de datos en las que se basan los sistemas de archivos modernos.

Pebibyte: Unidad de almacenamiento de información que corresponde a 2^{50} bytes.

Taxonomía: Ciencia que trata de los principios, métodos y fines de la clasificación.

Taxón: Unidad sistemática, como familia, género, etc.

WebDAV: Extensión del protocolo HTTP que facilita la colaboración entre usuarios para editar y gestionar documentos y ficheros almacenados en servidores de la *World Wide Web*.

2.Estado de la cuestión

En este capítulo se introducen los aspectos relacionados directa o indirectamente con el presente Trabajo Fin de Grado, tales como:

- Sistemas de ficheros
- Sistemas de ficheros semánticos
- Bases de Datos
- FUSE
- Python

2.1 Sistemas de ficheros

Los sistemas de ficheros o archivos, estructuran la información guardada en una unidad de almacenamiento (como por ejemplo un disco duro de una computadora, disquetes, discos ópticos, etc.), que después será representada utilizando un gestor de archivos. Estos sistemas proporcionan procedimientos para almacenar, recuperar y actualizar dichos datos, así como gestionar el espacio disponible en el dispositivo que lo contiene. Generalmente existe una estrecha relación entre el sistema operativo y el sistema de archivos, y la mayoría de sistemas operativos manejan su propio sistema de archivos. Algunos sistemas de ficheros proporcionan mecanismos para el control de acceso a los datos y metadatos. Asegurar la confiabilidad es una responsabilidad fundamental de un sistema de archivos.

Lo habitual es utilizar dispositivos de almacenamiento de datos que permiten el acceso a la información como una cadena de bloques de un mismo tamaño, a veces llamados sectores, usualmente de 512 bytes de longitud (también denominados *clusters*). El software del sistema de archivos es responsable de la organización de estos sectores en archivos y directorios y mantiene un registro de qué sectores pertenecen a qué archivos y cuáles no han sido utilizados. En la práctica, un sistema de archivos también puede ser utilizado para acceder a datos generados dinámicamente, como los recibidos a través de una conexión de red (sin la intervención de un dispositivo de almacenamiento).

El nombre de fichero se usa para referenciar la localización de la información en el sistema de ficheros. La mayoría de sistemas de ficheros tienen restricciones en la longitud del nombre. En algunos sistemas los nombres de fichero diferencian entre mayúsculas y minúsculas y en otros no. Existen sistemas de archivos en los que los nombres de fichero son estructurados, con sintaxis especiales para extensiones de archivos y números de versión. En otros, los nombres de archivos son simplemente cadenas de texto y los metadatos de cada archivo son alojados separadamente.

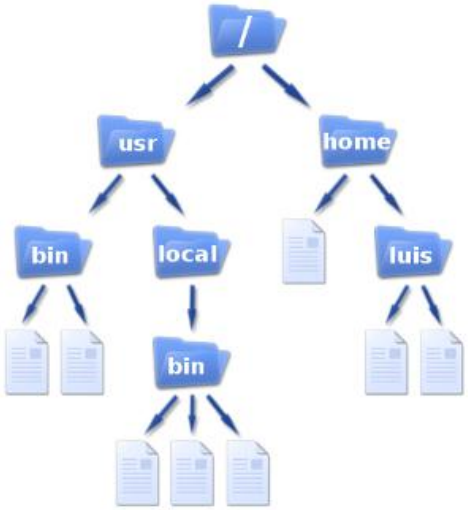

Figura 2.1: Sistema de ficheros jerárquico [4]

Normalmente los sistemas de ficheros tienen directorios (o carpetas) que permiten a los usuarios agrupar ficheros. Esto se implementa mediante la conexión del nombre de archivo a un índice en una tabla de contenido o a un i-nodo en sistemas de ficheros Unix. La estructura de los directorios suele ser jerárquica, ramificada o "en árbol", aunque en algún caso podría ser plana. En los sistemas de archivos jerárquicos, normalmente, se declara la ubicación precisa de un archivo con una cadena de texto llamada 'ruta'. La nomenclatura para rutas varía ligeramente de sistema en sistema, pero por lo general mantienen una misma estructura. Una ruta viene dada por una sucesión de nombres de directorios y subdirectorios, ordenados jerárquicamente de izquierda a derecha y separados por algún carácter especial (típicamente diagonal '/' o diagonal invertida '\') y puede terminar en el nombre de un archivo presente en la última rama de directorios especificada.

Los sistemas de ficheros incluyen utilidades para inicializar, modificar parámetros y borrar una instancia del sistema. Algunos incluyen la habilidad de extender o truncar el espacio reservado para el sistema de ficheros. Las utilidades de directorio son crear, renombrar y eliminar entradas de directorios y modificar metadatos asociados con un directorio. Las utilidades de fichero son crear, listar, copiar, mover y eliminar ficheros, y modificar metadatos.

Existen varios tipos de sistemas de ficheros:

- Sistemas de ficheros de disco: diseñados para el almacenamiento en una unidad de disco, que puede estar conectada directa o indirectamente a la computadora.

- Sistemas de ficheros de red: aquellos que acceden a sus archivos a través de una red. Dentro de esta clasificación encontramos dos tipos: los sistemas de archivos distribuidos y los sistemas de archivos paralelos.
- Sistemas de ficheros de propósito especial: aquellos tipos de sistemas de archivos que no son ni de disco ni de red. Ejemplos: cdfs, devfs, sysfs, wikifs, etc.

A continuación se exponen algunos de los sistemas de ficheros más conocidos actualmente.

FAT

FAT es un sistema de archivos desarrollado para MS-DOS, así como el sistema de archivos principal de las ediciones no empresariales de Microsoft Windows hasta Windows Me (las versiones posteriores tienen NTFS). Este sistema de archivos es relativamente sencillo pero robusto, lo que le convierte en un formato popular para disquetes admitido prácticamente por todos los sistemas operativos existentes para computadora personal. Además, FAT es el sistema de ficheros por defecto para los medios extraíbles (excepto CDs y DVDs) y como tal se encuentra normalmente en *discos floppy*, *super-floppies*, tarjetas de memoria y tarjetas de memoria flash, y la mayoría de dispositivos móviles lo soportan (PDAs, cámaras digitales, móviles, etc.).

El sistema de archivos FAT se caracteriza por la tabla de asignación de archivos (FAT). Esta tabla contiene entradas para cada *cluster* (área continua de almacenamiento en disco). Cuando se crea un archivo, se crea una entrada en el directorio y se establece el primer número de clúster que contiene datos. Esta entrada de la tabla FAT indica que éste es el último clúster del archivo o señala al clúster siguiente. La actualización de la tabla FAT es muy importante ya que si no se actualiza periódicamente, pueden producirse pérdidas de datos.

No hay ninguna organización en cuanto a la estructura de directorios de FAT y se asigna a los archivos la primera ubicación libre de la unidad.

Existen diferentes versiones de FAT, que indican el número de bits de cada elemento en la tabla: FAT12, FAT16 y FAT32 [5, 6].

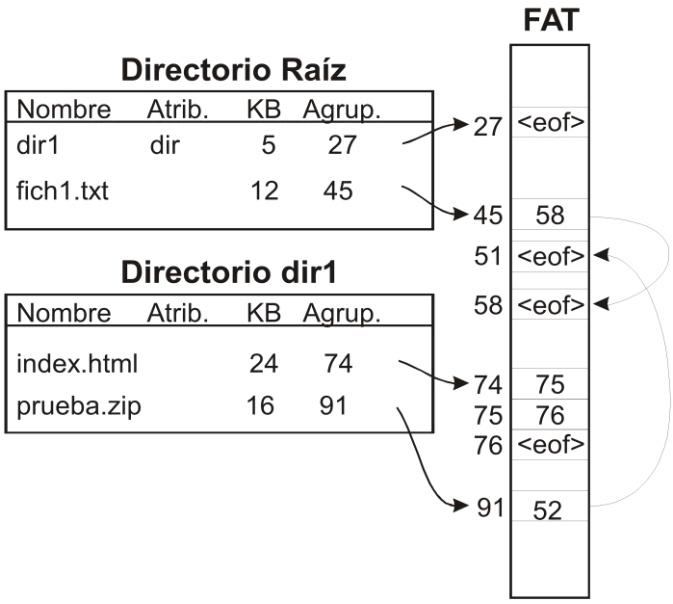

Figura 2.2: Estructura FAT [6]

NTFS

NTFS (*New Technology File System*) es el sistema de ficheros estándar de Windows NT, incluyendo Windows 2000, Windows XP y sus sucesores hasta la actualidad. NTFS sustituye a FAT como el sistema de archivos preferido por los sistemas operativos Microsoft Windows. Este sistema de ficheros tiene numerosas mejoras sobre FAT y HPFS, como perfeccionamiento del soporte para metadatos, y el uso de estructuras de datos avanzadas para mejorar el rendimiento, seguridad, y la utilización de espacio en disco.

Se basa en una estructura llamada "tabla maestra de archivos" o MTF, que puede contener información detallada en los archivos. Este sistema permite el uso de nombres extensos y distingue entre mayúsculas y minúsculas.

En cuanto al rendimiento, el acceso a los archivos en una partición NTFS es más rápido que en una partición de tipo FAT, ya que usan un árbol binario de alto rendimiento para localizar a los archivos [5].

Ext

El sistema de ficheros extendido o 'ext' fue el primer sistema de ficheros creado específicamente para el *kernel* de Linux. Posee una estructura basada en metadatos, inspirada en el sistema de ficheros tradicional de Unix (UFS). Fue la primera implementación que utilizó el sistema de ficheros virtual (VFS) [7].

Este sistema de ficheros fue el primero de la familia de sistemas de ficheros extendidos. Cada miembro de la familia introduce mejoras a la versión anterior. El resto de componentes de la familia de ficheros extendidos son:

- ext2

  El espacio en ext2 está dividido en bloques, que están agrupados en grupos de bloques. Típicamente hay miles de bloques en un sistema de ficheros grande. La información de un fichero está normalmente contenida dentro de un único grupo de bloques si es posible, para reducir el número de discos en los que se busca cuando se lee grandes cantidades de datos consecutivos.

  Cada archivo o directorio está representado por un i-nodo. El i-nodo incluye datos sobre el tamaño, permisos, propiedad y localización en el disco del archivo o directorio. El sistema de ficheros tiene una tabla donde se almacenan dichos i-nodos.

- ext3

  Esta versión del sistema de ficheros añade a la anterior: registro por diario o *Journaling*, índices en árbol para directorios que ocupan múltiples bloques y crecimiento en línea.

- ext4

  Las principales mejoras que introduce ext4 respecto a versiones anteriores son: soporte de volúmenes de hasta 1024 PiB, soporte añadido de *extent*, menor uso de la CPU y mejoras en la velocidad de lectura y escritura.

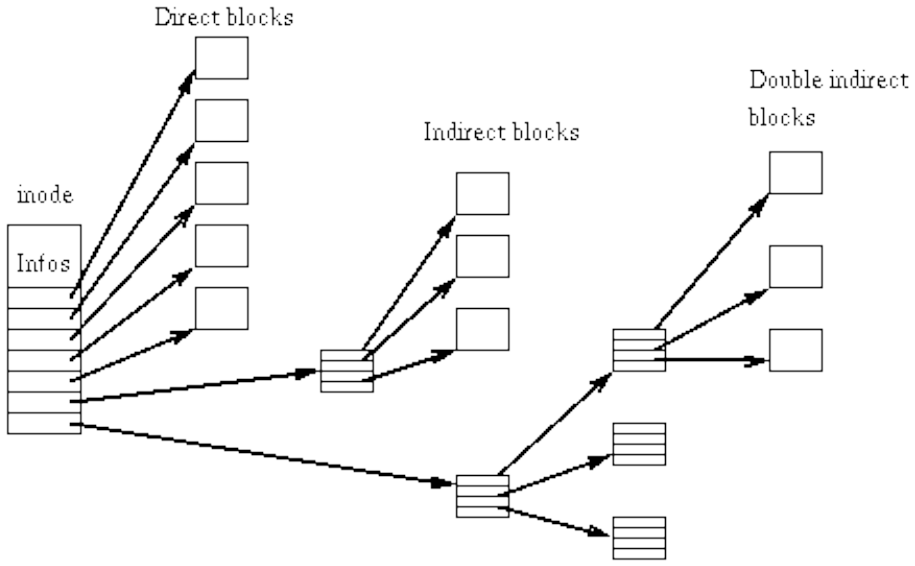

Figura 2.3: Estructura de i-nodos y bloques ext2 [7]

2.2 Sistemas de Ficheros Semánticos

Los sistemas de ficheros semánticos son sistemas de ficheros utilizados para la persistencia de la información, estructurando los datos de acuerdo a su semántica y propósito. Permiten ordenar y recuperar los datos por su contenido [8].

Este tipo de sistemas de ficheros basa su organización en los metadatos de los archivos, posibilitando la clasificación y localización de estos en base a diferentes tipos de información (tamaño, fecha de creación, temática tratada, etc.). Para ello se suele utilizar la arquitectura mostrada en la Figura 2.4, en la que el almacenamiento físico se produce en el sistema de ficheros tradicional, y sin embargo al usuario se le muestra una clasificación semántica [2, 3].

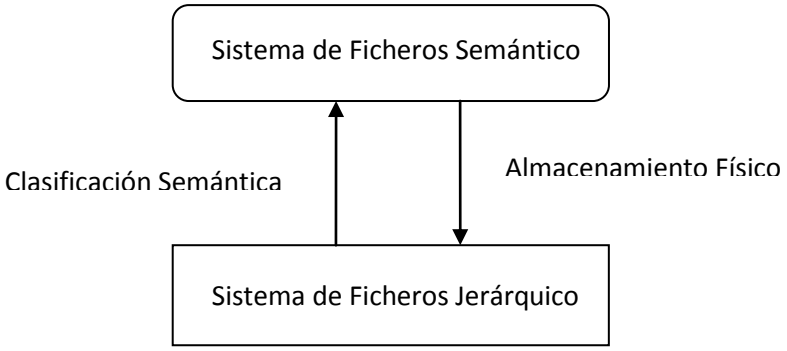

Figura 2.4: Arquitectura Sistemas de Ficheros Semánticos

A continuación se explican algunos de estos sistemas existentes.

SemFS

SemFS es un sistema de ficheros semánticos basados en RDF. En Windows puede montarse como un driver *WebDAV*. En Linux, SemFS puede usarse a nivel de sistema de ficheros de usuario vía FUSE. Actualmente SemFS soporta etiquetado de archivos y navegación conforme a diferentes ontologías.

SemFS reinterpreta el concepto de recursos del sistema de archivos y sus localizaciones como recursos y anotaciones de recursos en bases de conocimiento RDF. Accediendo a conocimiento estructurado, como información bibliográfica o información de contactos a través de archivos, este enfoque ofrece la posibilidad de integrar el contenido de las bases de conocimiento con los sistemas de escritorio de usuario.

Específicamente, SemFS proporciona un repositorio de recursos en red donde los recursos son almacenados y accedidos a través del paradigma estándar de sistema de ficheros jerárquico

aplicado en los metadatos del recurso en lugar del nombre del directorio en el que se encuentra el archivo. La lógica real del sistema de ficheros está implementada en dos componentes:

- El componente *ClassHandler*: responsable de presentar como archivos los recursos en la base de conocimiento.
- El componente *Filter*: responsable de seleccionar el conjunto apropiado de archivos a mostrar en un directorio dado.

Esta distinción es potente, ya que permite diseñar y operar *ClassHandlers* y *Filters* independientemente unos de otros [9].

Tagxfs

Tagxfs [10] es un sistema de archivos semántico. Extiende el sistema de ficheros de espacio de usuario a una jerarquía basada en etiquetas, enfocada facilitar las búsquedas.

La extensión que proporciona Tagxfs se compone de enlaces, etiquetas y una jerarquía de nodos. Las etiquetas representan categorías únicas y pueden ser de dos tipos:

- Etiqueta de filtro: asocia a los archivos una categoría específica (ciencia, aventura)
- Etiqueta grupal: agrupa a los archivos por una propiedad (género, año)

Los enlaces son una representación simbólica de la carpeta que queremos categorizar. Están representados por un único nombre corto y un *path* completo en el sistema de archivos de base. Es posible marcar los enlaces con una serie de etiquetas, después usadas como filtros en la jerarquía de nodos.

La jerarquía de nodos es la representación de las etiquetas en una carpeta con estructura en árbol. Se compone de nodos (nodos jerárquicos), y hojas (enlaces). Para cada nodo se puede configurar una etiqueta, una serie de sub-nodos, un modo y una posición única en la jerarquía de nodos existentes. El modo de la jerarquía de nodos define qué enlaces se listarán. Los modos disponibles son:

- *All*: lista los sub-nodos y todos los enlaces disponibles en la jerarquía debajo del nodo escogido. No se aplican criterios de filtro.
- *Links*: lista los sub-nodos y los enlaces filtrados por las etiquetas indicadas
- *None*: sólo lista los sub-nodos

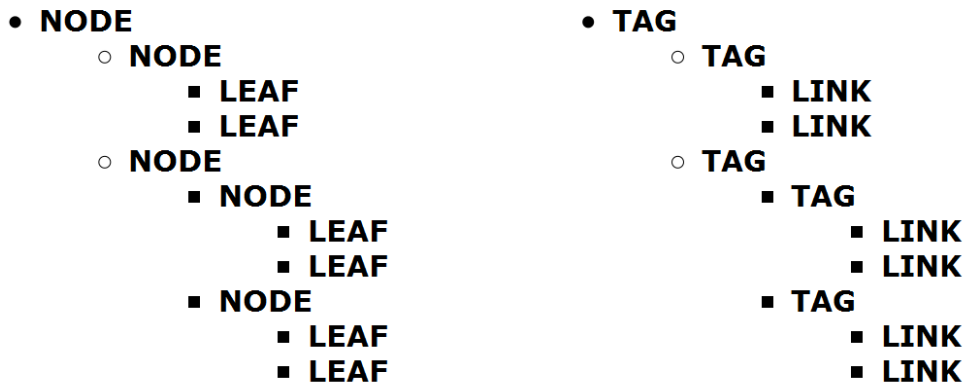

Figura 2.5: Jerarquía de Nodos Tagxfs [10]

WinFS

WinFS (*Windows Future Storage*) es el nombre de un proyecto cancelado de un sistema de almacenamiento y gestión de datos. Está basado en bases de datos relacionales y fue desarrollado por Microsoft [11].

Este sistema de archivos reconoce diferentes tipos de datos, como fotografías, e-mail, documentos, audio, video, calendarios, contactos, etc; en lugar de tratarlos como un flujo de bytes sin analizar (como hacen la mayoría de sistemas de ficheros). Los datos almacenados y gestionados por el sistema son instancias de los datos reconocidos por WinFS. Los datos son estructurados según sus propiedades. Cada propiedad puede ser de tipo simple, como *String*, *Integer*, o *Date,* o de tipo complejo como *Contact*. En adición, WinFS permite vincular dos instancias de datos diferentes, como un contacto y un documento que pueden ser asociados por una relación de 'Creado por'.

Las asociaciones también se tratan como propiedades, de tal forma que si un documento está asociado a un contacto por una relación 'Creado por', el documento tendrá la propiedad 'Creado por'. Cuando se accede al documento, se atraviesa la asociación y se devuelven los datos relacionados.

El acceso a todos los datos en el sistema permite búsquedas complejas (varios parámetros) que se realizarán a través de los ítems de datos gestionados por WinFS. En caso de querer realizar una búsqueda de esta características como 'El número de teléfono de todas las personas que viven en Acapulco y tengan más de 100 apariciones en mi colección de fotos y con los que haya tenido contacto por e-mail el último mes'. WinFS puede encontrar la propiedad 'sujeto' de todas las fotos y atravesar su asociación para encontrar los ítems de

contacto. Puede filtrar los emails en el último mes y acceder a la asociación 'Comunicado con' para alcanzar los contactos. Ambas listas de contactos resultantes se comparan y se obtienen los números de teléfono buscados.

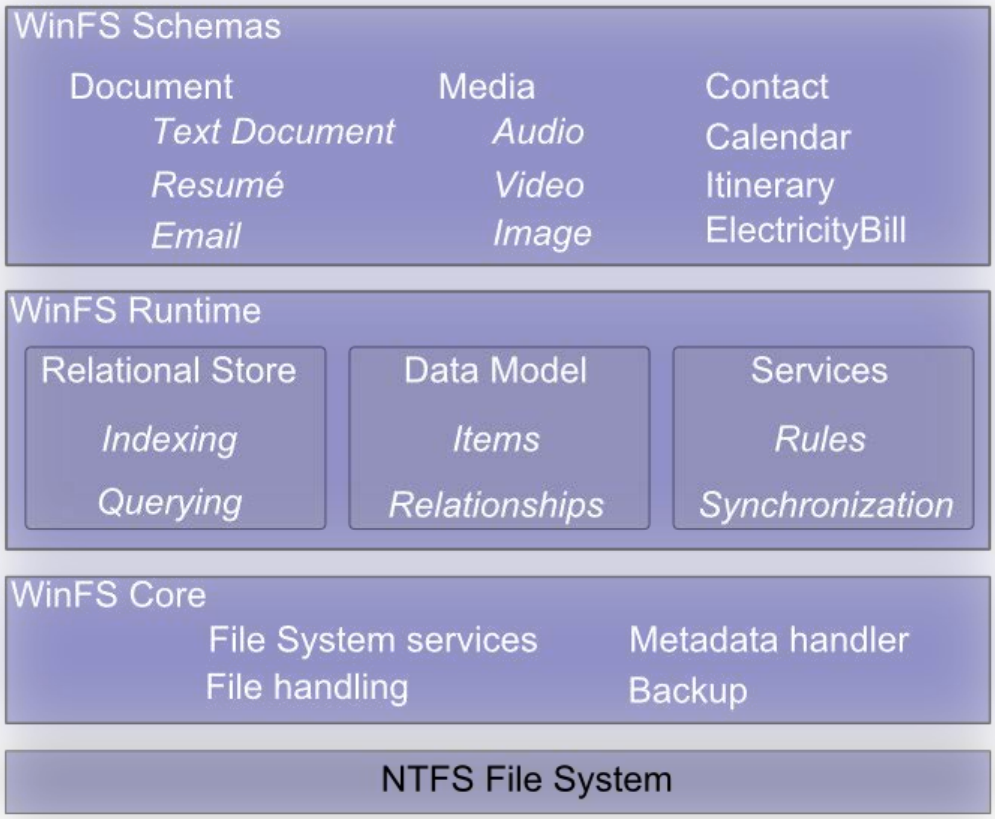

Figura 2.6: Arquitectura de WinFS [12]

2.3 Bases de datos

Una base de datos es un sistema de recopilación de información. Una de sus funciones básicas es el almacenamiento de información y una de sus principales utilidades es la de encontrar dicha información de manera rápida y sencilla. De este modo, una buena base de datos será aquella que permita introducir todo tipo de información con rapidez y facilidad y, a la vez, facilite la recuperación inmediata de toda o parte de esta información mediante criterios de búsqueda cuya definición resulte sencilla para el usuario.

De una forma simple, una base de datos puede ser definida como un conjunto de información organizada, agrupada y estructurada con un propósito particular. En informática, una base de datos es un sistema formado por un conjunto de datos almacenados en soportes de acceso aleatorio que permiten el acceso directo a ellos y un conjunto de programas, denominados sistemas de gestión de bases de datos o SGBD, que administran y manipulan dichos datos. Desde un punto de vista más formal, se puede definir una base de datos como un conjunto de datos estructurados, fiables y homogéneos, organizados de forma independiente de la máquina donde se almacenan, accesibles en tiempo real, compartibles por usuarios concurrentes con necesidades de información diferentes y no predecibles en el tiempo [13].

Figura 2.7: Principales Sistemas de Gestión de Bases de Datos Relacionales [14]

2.3.1 Bases de Datos Relacionales

La mayoría de SGBD utilizan un modelo de gestión de bases de datos relacional para almacenar la información. Los sistemas relacionales ofrecen facilidad de uso para el usuario final, aprendizaje relativamente corto y sencillez en la consulta de información.

En un sistema de gestión de bases de datos relacional, el sistema trata todos los datos en tablas. En una tabla se almacena, de forma organizada, toda la información homogénea referente a un tema. Habitualmente la información está contenida en varias tablas relacionadas entre sí mediante campos comunes, lo que permite evitar la información redundante.

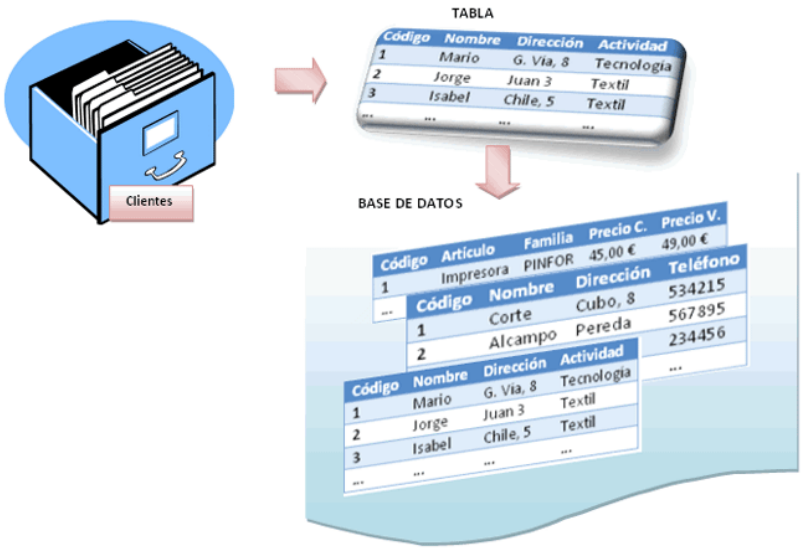

Figura 2.8: Tablas en una Base de Datos relacional [15]

Las tablas están formadas por registros (filas) que recogen toda la información de un determinado elemento. Los registros están formados por campos (columnas) que contienen un dato concreto de cada elemento de la tabla.

Las bases de datos relacionales cumplen las siguientes leyes básicas:

- Están formadas por más de una tabla.
- Cada una de las tablas tiene un nombre diferente, cuyo primer carácter será alfabético y que no podrá contener caracteres especiales.
- Cada registro de una tabla contiene un número de campos fijo.

- El nombre de todos los campos de una tabla es distinto, por lo que se pueden utilizar nombres iguales para campos de tablas distintas.
- El orden de los registros y de los campos no está determinado, sino es aleatorio.

Los sistemas de gestión de base de datos SQL más utilizados son MySQL, Oracle y PostgreSQL.

MySQL

MySQL es un sistema gestor de bases de datos SQL. Es de código abierto, desarrollado y distribuido por Oracle.

Este sistema gestor de bases de datos relacionales, ofrece compatibilidad con PHP, Perl, C y HTML, y funciones avanzadas de administración y optimización de bases de datos para facilitar las tareas habituales. Implementa funciones Web, permitiendo un acceso seguro y sencillo a los datos a través de Internet. MySQL incluye capacidades de análisis integradas, servicios de transformación y duplicación de datos y funciones de programación mejoradas.

Se puede decir que MySQL es un sistema cliente servidor de administración de bases de datos relacionales diseñado para el trabajo tanto en los sistemas operativos Windows como en los sistemas UNIX/LINUX. En adición, determinadas sentencias de MySQL pueden ser embebidas en código PHP y HTML para diseñar aplicaciones Web dinámicas que incorporan la información de las tablas de MySQL a páginas Web. Así mismo, MySQL es compatible con el software más potente de diseño Web, como *Dreamweaver MS* [16].

2.3.2 Bases de Datos NoSQL

Amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales en aspectos importantes. El más destacado es que no usan SQL como el principal lenguaje de consultas (de ahí su nombre). Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente la atomicidad, coherencia, aislamiento ni durabilidad, y habitualmente escalan bien horizontalmente.

Las bases de datos NoSQL están altamente optimizadas para las operaciones recuperar y agregar.

Existen varios tipos de bases de datos NoSQL:

- Documentales: se dedicará un apartado a su explicación.
- En grafo: representan la información como nodos de un grafo y sus relaciones con las aristas del mismo, de manera que se pueda usar teoría de grafos para recorrer la base de datos.
- Clave/valor: formadas por contenedores (o *cabinets*) en los que pueden existir tantas parejas de clave-valor como sea necesario. En cada contenedor puede haber datos de la misma naturaleza o totalmente diferentes.
- Orientadas a objetos: la información se representa mediante objetos, como los presentes en la programación orientada a objetos.
- Otros: multivalor, tabular, etc.

Bases de Datos Documentales

En este tipo de Bases de Datos, cada registro se corresponde con un documento, sea éste de cualquier tipo. Permiten la indexación a texto completo, y en líneas generales realizan búsquedas más potentes. Un SGBDD (Sistema de Gestión de Bases de Datos Documentales) se ocupa de la gestión de documentos optimizando el almacenaje y facilitando su recuperación. A diferencia de cualquier otro SGBD, un SGBDD no realiza ningún tratamiento sobre la información, simplemente la almacena y posibilita su recuperación, lo que agiliza enormemente la búsqueda y recuperación de los datos.

Los principales Sistema de Gestión de Bases de Datos Documentales son CouchDB y MongoDB.

Mongo DB

MongoDB [17] es un sistema de bases de datos *NoSQL*. Es multiplataforma y orientado a documentos. En lugar de almacenar los datos en tablas como se hace en una Base de Datos clásica relacional, *MongoDB* almacena información estructurada como documentos JSON (en *MongoDB BSON* o *Binary JSON*) con esquemas dinámicos (cada entrada o registro puede tener un esquema de datos diferentes), de tal forma que la integración y recuperación de datos se facilita y agiliza enormemente.

Es un sistema de bases de datos escalable, de alto rendimiento y *open source*, escrito en C++. Su nombre viene del término "hu**mongo**us" (enorme, colosal).

En *MongoDB*, cada registro o conjunto de datos se denomina documento. Los documentos se pueden agrupar en colecciones, lo que sería el equivalente a las tablas en una base de datos

relacional. La diferencia es que en las colecciones se pueden almacenar documentos con muy diferentes formatos, en lugar de estar sometidos a un esquema fijo.

Las características principales del mencionado sistema de bases de datos son:

- Consultas Ad hoc: *MongoDB* soporta búsquedas por campo, consultas de rango y búsquedas con expresiones regulares. Las consultas pueden devolver campos específicos de los documentos, pero también pueden ser funciones *JavaScript* definidas por el usuario.
- Indexado: Se pueden crear índices para cualquier campo de los documentos, de modo que *MongoDB* mantiene una estructura interna eficiente para el acceso a la información por los contenidos de estos campos. También soporta índices secundarios.
- Replicación: *MongoDB* soporta replicación maestro-esclavo. El maestro puede realizar lecturas y escrituras. Un esclavo copia los datos del maestro, que puede usar exclusivamente para leer o hacer *backup* (no escribir). Los esclavos tienen la habilidad de elegir un nuevo maestro si el servicio con el maestro actual se cae.
- Balanceo de carga: *MongoDB* escala horizontalmente usando *sharding*. El desarrollador elige una clave *shard*, que determina como se distribuyen los datos en una colección. La información es dividida en rangos y distribuida a través de múltiples *shards* (un *shard* es un maestro con uno o más esclavos). *MongoDB* puede ejecutarse sobre múltiples servidores, balanceando la carga y/o duplicando los datos para mantener el sistema ejecutando en caso de que se produzcan fallos de hardware.
- Almacenamiento de ficheros: *MongoDB* puede utilizarse como un sistema de ficheros, aprovechando las características de balanceo de carga y replicación de datos.
- Agregación: es posible el procesamiento por lotes de datos y operaciones de agregación. *MongoDB* permite a los usuarios obtener el tipo de resultado que se obtiene cuando se utiliza el comando SQL “group-by”.

2.4 FUSE

FUSE es un módulo cargable en el núcleo para sistemas operativos tipo Unix, que permite a usuarios no privilegiados crear sus propios sistemas de archivos sin necesidad de editar el código del núcleo. Esto se logra mediante la ejecución del código del sistema de archivos en el espacio de usuario, mientras que el módulo FUSE sólo proporciona un "puente" a la interfaz del núcleo real. FUSE fue oficialmente integrado en el *kernel* de Linux en la versión 2.6.14.

FUSE es realmente útil para la creación de sistemas de archivos virtuales. A diferencia de los tradicionales sistemas de archivos, que, en esencia, guardan y recuperan los datos desde un disco, los sistemas de archivos virtuales en realidad no almacenan datos propios. Actúan como una visualización o traducción de un sistema de archivos existente o dispositivo de almacenamiento.

Cuando se monta un programa FUSE sobre un directorio todas las llamadas que se realicen sobre este directorio son derivadas a este programa, que es el encargado de devolver el valor de la llamada.

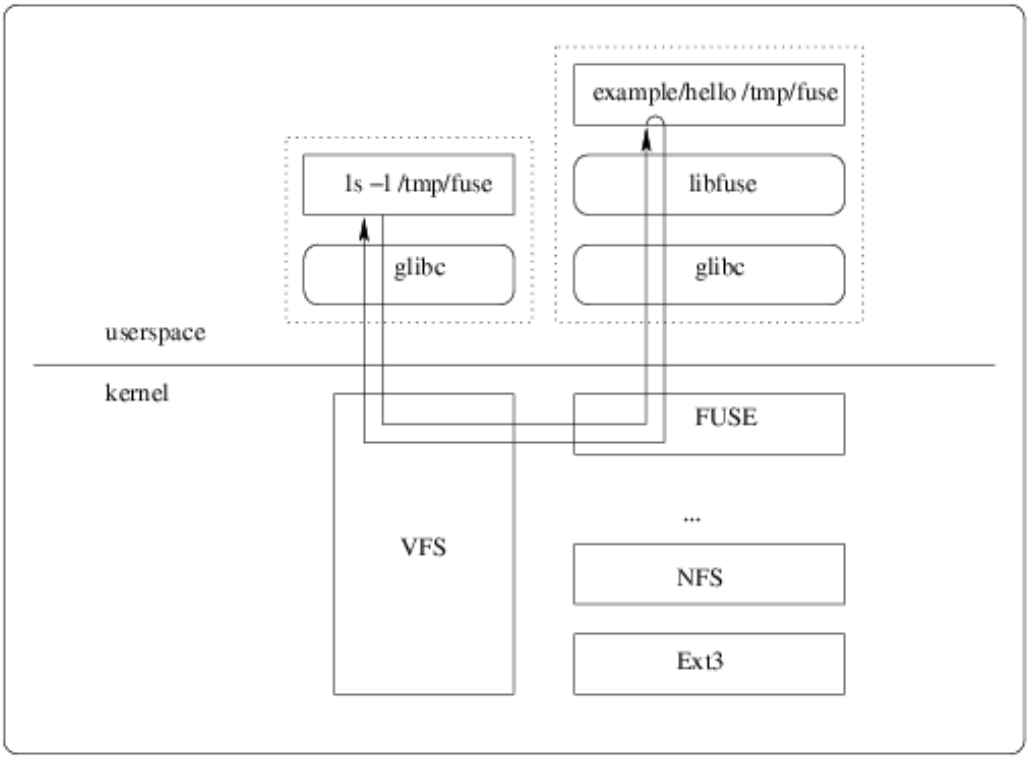

Figura 2.9: Funcionamiento de FUSE [18]

Proyectos realizados con FUSE

Existen numerosos sistemas de ficheros realizados con FUSE. A continuación se exponen algunos de ellos, divididos por tipo.

Sistemas de Ficheros con Bases de Datos

Dentro de los proyectos que hacen uso de una base de datos se pueden dividir en 3 grandes grupos:

- Uso de la base de datos para el almacenamiento de metadatos. Como por ejemplo el proyecto Tagsistant que asocia etiquetas a los ficheros actuales del sistema.
- Uso de la base de datos para el almacenamiento puro de datos.
- Uso mixto de la base de datos. Como el proyecto CopyFS que almacena los datos de los ficheros y a su vez metadatos del tipo fecha y hora de creación y modificación.

Sistemas de Ficheros de Red

Su propósito es almacenar archivos en equipos remotos como servidores de archivos y sitios web.

- SSHFS: Este es un cliente del sistema de archivos basado en el protocolo de transferencia de archivos SSH. Como la mayoría de los servidores SSH ya son compatibles con este protocolo es muy fácil de configurar: es decir, del lado del servidor no hay nada que hacer. Por el lado del cliente el montaje del sistema de archivos es tan fácil como conectarse al servidor con SSH.
- HTTPFS: monta cualquier archivo accesible mediante http, en modo lectura, el servidor debe soporta HTTP/1.1.

Otros

- GlusterFS: Sistema de archivos agrupado distribuido con capacidad de petabytes.
- GmailFS: Sistema de archivos que guarda los datos como correos en Gmail
- WikipediaFS: Muestra y edita los artículos de Wikipedia como si fueran archivos

2.5 Python

Python es un lenguaje de programación de alto nivel, que fue creado a principios de los años 90 por Guido van Rossum, cuyo nombre está inspirado en el grupo de cómicos ingleses "*Monty Python*". Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible [19].

Se trata de un lenguaje interpretado (o de script), con tipado dinámico, fuertemente tipado y multiplataforma. Es multiparadigma ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

Lenguaje interpretado o de script

Un lenguaje interpretado o de script es aquel que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora (lenguajes compilados).

La ventaja de los lenguajes compilados es que su ejecución es más rápida. Sin embargo los lenguajes interpretados son más flexibles y más portables.

Python tiene, no obstante, muchas de las características de los lenguajes compilados, por lo que se podría decir que es semi-interpretado. En Python, como en Java y muchos otros lenguajes, el código fuente se traduce a un pseudocódigo máquina intermedio llamado *bytecode* la primera vez que se ejecuta, generando archivos .pyc o .pyo (*bytecode* optimizado), que son los que se ejecutarán en sucesivas ocasiones.

Tipado dinámico

La característica de tipado dinámico se refiere a que no es necesario declarar el tipo de dato que va a contener una determinada variable, sino que su tipo se determinará en tiempo de ejecución según el tipo del valor al que se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo.

Fuertemente tipado

No se permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente. Por ejemplo, si tenemos una variable que contiene un texto (variable de tipo cadena o *String*) no podremos tratarla como un número (sumar la cadena "9" y el número 8). En otros lenguajes el tipo de la variable

cambiaría para adaptarse al comportamiento esperado, aunque esto es más propenso a errores.

Multiplataforma

El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios.

Orientado a objetos

La orientación a objetos es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se trasladan a clases y objetos en nuestro programa. La ejecución del programa consiste en una serie de interacciones entre los objetos.

Programación imperativa

La programación imperativa es otro paradigma de programación que describe la programación en términos del estado del programa y sentencias que cambian dicho estado. Los programas imperativos son un conjunto de instrucciones que le indican al computador cómo realizar una tarea.

3.Marco Regulador

Este capítulo recoge las normativas técnicas y legales que afectan al trabajo.

Dado que el sistema aquí tratado es un sistema de ficheros, y su principal objetivo es la organización y localización de datos, es necesario contemplar la ley de Protección de Datos de Carácter Personal (Ley Orgánica 15/1999, de 13 de diciembre).

En el Artículo 2, Ámbito de aplicación, párrafo 2 se dice:

El régimen de protección de los datos de carácter personal que se establece en la presente Ley Orgánica no será de aplicación:

a) *A los ficheros mantenidos por personas físicas en el ejercicio de actividades exclusivamente personales o domésticas.*
b) *A los ficheros sometidos a la normativa sobre protección de materias clasificadas.*
c) *A los ficheros establecidos para la investigación del terrorismo y de formas graves de delincuencia organizada. No obstante, en estos supuestos el responsable del fichero comunicará previamente la existencia del mismo, sus características generales y su finalidad a la Agencia Española de Protección de Datos.*

El sistema de ficheros tratado en el presente documento tiene como fin ser utilizado en un ámbito personal, siendo los datos gestionados por el propio usuario. Por lo tanto dicha ley no es aplicable en este caso.

4. Análisis y Diseño

Este capítulo contiene el análisis y diseño del sistema de ficheros CollectionFS, así como la metodología utilizada para dicho análisis y diseño.

4.1 Metodología

El objetivo principal del presente Trabajo Fin de Grado es el análisis y diseño de un sistema de ficheros semántico. Para validar dicho análisis y diseño, se realizará el desarrollo de un prototipo ágil que cumpla con la funcionalidad básica del sistema analizado. En la elección de la metodología a utilizar, se deben tener en cuenta las características de dicho prototipo: ágil y orientado a la validación del análisis y diseño. Por ello se va a optar por utilizar una metodología ágil.

Las metodologías ágiles tienen como propósito permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se basan en el manifiesto ágil [20].

Existen numerosas metodologías ágiles, la que se va a utilizar en el presente TFG es la denominada "*Adaptative Software Development*". Sus principales características son: iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda se desarrollan las características y finalmente en la tercera se revisa su calidad. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.

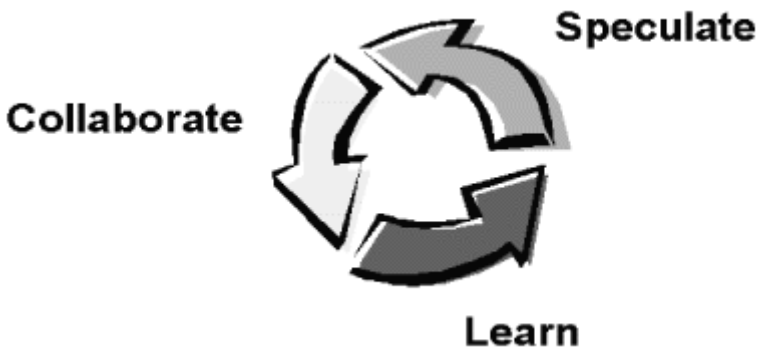

Figura 4.1: Fases Metodología [21]

4.1.1 Fases

La metodología *Adaptative Software* Development divide el desarrollo de un proyecto en tres fases: Planificación, Construcción y Revisión. Para el presente trabajo solo se realizará una iteración. Las tareas que se realizarán en cada fase de dicha iteración se muestran a continuación.

Especulación

En esta fase se inicia el proyecto, por lo tanto es necesario definir los objetivos y realizar la planificación. Así mismo se realizará un aprendizaje de las tecnologías a utilizar.

- Definir objetivos
- Aprendizaje de las tecnologías a utilizar en el proyecto: debido a que se utilizarán tecnologías que el equipo de desarrollo no conoce, es necesaria la familiarización con dichas tecnologías en esta primera fase.
- Planificar tiempo: el resultado de la planificación se recoge en el diagrama de Gantt, Anexo VIII.

Colaboración

En esta fase se llevan a cabo los objetivos marcados en el inicio del trabajo.

- Análisis: se realizará un análisis del sistema de ficheros semántico propuesto en el presente TFG. Para ello se analizarán los casos de uso y se obtendrán los requisitos y el modelo de la base de datos.
- Diseño: se realizará el diseño del sistema, teniendo en cuenta las diferentes opciones de diseño.
- Desarrollo: finalmente se desarrollará un prototipo del sistema analizado y diseñado.

Aprendizaje

En esta fase se realiza la revisión del proyecto construido. Para ello se realiza un análisis de rendimiento evaluando las características finales del producto. Una vez realizada dicha evaluación se proponen mejoras y se sacan conclusiones.

- Análisis de Rendimiento: evaluación del rendimiento del sistema. Se realizarán pruebas de inserción, búsqueda y borrado para medir la eficiencia del sistema.
- Conclusiones: se analizará si los objetivos propuestos han sido cumplidos.
- Mejoras a realizar (trabajos futuros): una vez realizada la revisión se podrán proponer mejoras a realizar en posibles trabajos futuros (ya que el TFG se limita al desarrollo de un prototipo).

4.2 Análisis

En este apartado se expone el análisis realizado. Dicho análisis está formado por los casos de uso del sistema de ficheros CollectionFS, los requisitos obtenidos a partir de dichos casos de uso, y el modelo entidad relación de la base de datos del sistema.

4.2.1 Casos de Uso

En este apartado se muestra una representación de la funcionalidad del sistema a analizar, diseñar y desarrollar.

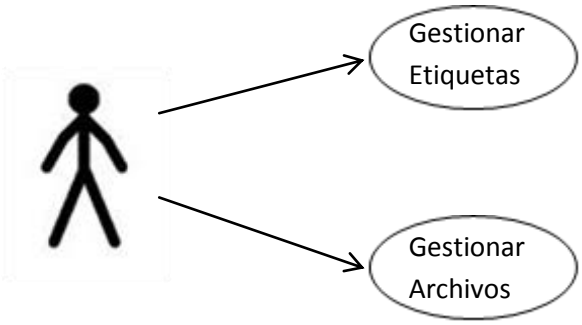

Figura 4.2: Diagrama de Casos de Uso

Como se muestra en la Figura 4.2, existen dos Casos de Uso generales: Gestionar Etiquetas y Gestionar Archivos. A continuación se muestra cada unos de estos casos de uso desglosado (Figuras 4.3 y 4.4).

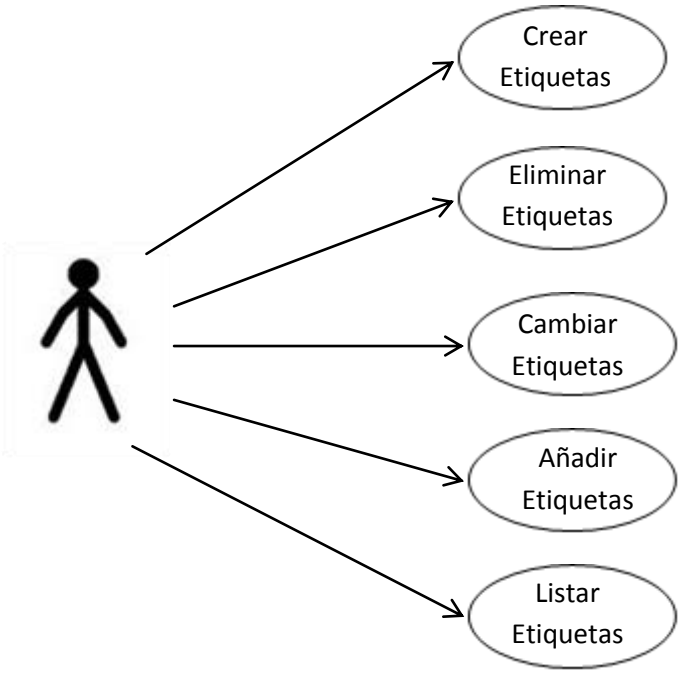

Figura 4.3: Diagrama de Casos de Uso Gestionar Etiquetas

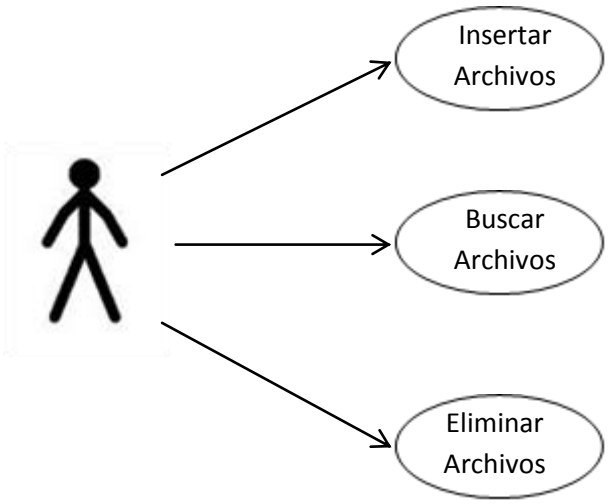

Figura 4.4: Diagrama de Casos de Uso Gestionar Archivos

4.2.2 Requisitos

Tras el análisis de los Casos de Uso que se expone en la sección anterior, se han obtenido los requisitos del sistema. Dichos requisitos se dividen en Requisitos de Usuario y Requisitos de Software.

4.2.2.1 Requisitos de Usuario

Los requisitos de Usuario definen lo que el usuario será capaz de realizar en el sistema. Para su correcta definición, cada requisito dispondrá de los siguientes campos:

- Nombre: compuesto por las iniciales 'RU' seguidas del número de requisito, y un título breve del mismo.
- Descripción: contiene una explicación más amplia del requisito.
- Tipo: Capacidad (lo que el usuario puede hacer) o Restricción (lo que no puede hacer).
- Necesidad: describe si el requisito es esencial u opcional.
- Prioridad: describe la prioridad de implementación del requisito. Posibles valores: Baja, Media y Alta
- Verificabilidad: describe en qué medida el requisito es verificable. Posibles valores: Baja, Media y Alta

RU01: Inicializar sistema			
Descripción	El usuario podrá inicializar el sistema, creando la configuración inicial.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.1: Requisito RU01

RU02: Crear Etiquetas			
Descripción	El usuario podrá crear nuevas etiquetas.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.2: Requisito RU02

RU03: Crear etiquetas por defecto			
Descripción	El usuario no podrá crear etiquetas dentro de las etiquetas por defecto 'Extensión' y 'Tamaño'.		
Tipo	Restricción	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.3: Requisito RU03

RU04: Crear más de 2 niveles			
Descripción	El usuario no podrá crear más de dos niveles de etiquetas.		
Tipo	Restricción	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.4: Requisito RU04

RU05: Eliminar Etiquetas			
Descripción	El usuario podrá eliminar las etiquetas creadas por él.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.5: Requisito RU05

RU06: Eliminar Etiquetas por Defecto			
Descripción	El usuario no podrá eliminar las etiquetas por defecto: • Extensión • Tamaño		
Tipo	Restricción	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.6: Requisito RU06

RU07: Cambiar etiqueta			
Descripción	El usuario podrá cambiar una etiqueta de un archivo por otra.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.7: Requisito RU07

RU08: Añadir etiqueta			
Descripción	El usuario podrá añadir una etiqueta a un archivo.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.8: Requisito RU08

RU09: Eliminar etiqueta de archivo			
Descripción	El usuario podrá suprimir la asociación entre una etiqueta y un archivo, es decir, podrá eliminar una etiqueta de un archivo.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.9: Requisito RU09

RU10: Eliminar etiqueta por defecto de archivo			
Descripción	El usuario no podrá suprimir la asociación entre una etiqueta por defecto y un archivo, es decir, no podrá eliminar las etiquetas por defecto de un archivo.		
Tipo	Restricción	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.10: Requisito RU10

RU11: Listar etiquetas			
Descripción	El usuario podrá listar las etiquetas disponibles en el sistema.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.11: Requisito RU11

RU12: Insertar Archivos			
Descripción	El usuario podrá insertar archivos en el sistema.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.12: Requisito RU12

RU13: Cambiar nombre			
Descripción	El usuario podrá cambiar el nombre de los archivos insertados en el sistema.		
Tipo	Capacidad	Necesidad	Opcional
Prioridad	Baja	Verificabilidad	Alta

Tabla 4.13: Requisito RU13

RU14: Añadir Descripción			
Descripción	El usuario podrá añadir una descripción a los archivos insertados en el sistema. Podrá cambiar dicha descripción siempre que desee.		
Tipo	Capacidad	Necesidad	Opcional
Prioridad	Baja	Verificabilidad	Alta

Tabla 4.14: Requisito RU14

RU15: Buscar por nombre			
Descripción	El usuario podrá buscar archivos insertados en el sistema mediante el nombre.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.15: Requisito RU15

RU16: Buscar por extensión			
Descripción	El usuario podrá buscar archivos por su extensión.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.16: Requisito RU16

RU17: Buscar por tamaño			
Descripción	El usuario podrá buscar ficheros por tamaño.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.17: Requisito RU17

RU18: Buscar por etiqueta			
Descripción	El usuario podrá buscar archivos por las etiquetas que él mismo ha asignado.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.18: Requisito RU18

RU19: Buscar por md5			
Descripción	El usuario podrá realizar búsquedas de archivos por su md5.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.19: Requisito RU19

RU20: Búsquedas combinadas			
Descripción	El usuario podrá realizar búsquedas combinadas con uno, varios o todos los siguientes campos: • Nombre • Extensión • Tamaño • Etiquetas • Md5		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.20: Requisito RU20

RU21: Eliminar archivos			
Descripción	El usuario podrá eliminar archivos completamente del sistema.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.21: Requisito RU21

RU22: Abrir archivos			
Descripción	El usuario podrá abrir los archivos insertados en el sistema. Al realizar una búsqueda, el usuario será capaz de abrir el archivo encontrado.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.22: Requisito RU22

RU23: Insertar Programas			
Descripción	El usuario podrá insertar en la configuración del sistema los programas con los que desea abrir las diferentes extensiones de archivo.		
Tipo	Capacidad	Necesidad	Esencial
Prioridad	Baja	Verificabilidad	Alta

Tabla 4.23: Insertar Programas

Matriz Trazabilidad: Casos de Uso – Requisitos de Usuario

	CU1: Crear Etiquetas	CU2: Eliminar Etiquetas	CU3: Cambiar Etiqueta	CU4: Añadir Etiqueta	CU5: Listar Etiquetas	CU6: Insertar Archivos	CU7: Buscar Archivos	CU8: Eliminar Archivos
RU01	X	X	X	X	X	X	X	X
RU02	X							
RU03	X							
RU04	X							
RU05		X						
RU06		X						
RU07			X					
RU08				X				
RU09		X						
RU10		X						
RU11					X			
RU12						X		
RU13						X		
RU14						X		
RU15							X	
RU16							X	
RU17							X	
RU18							X	
RU19							X	
RU20							X	
RU21								X
RU22							X	
RU23							X	

Tabla 4.24: Matriz Trazabilidad Casos de Uso – Requisitos de Usuario

4.2.2.2 Requisitos de Software

Los requisitos de software describen la funcionalidad que deberá tener el sistema. Para su correcta definición, los requisitos se componen de los siguientes campos:

- Nombre: compuesto por las iniciales 'RU' seguidas del número de requisito.
- Descripción: contiene una explicación más amplia del requisito.
- Tipo: indica el tipo del requisito de software.
  - Funcional: define el comportamiento del software
  - No funcional: impone restricciones en el diseño o la implementación
- Necesidad: describe si el requisito es esencial u opcional.
- Prioridad: describe la prioridad de implementación del requisito. Posibles valores: Baja, Media y Alta
- Verificabilidad: describe en qué medida el requisito es verificable. Posibles valores: Baja, Media y Alta

RS01			
Descripción	El sistema debe permitir al usuario inicializar el sistema y realizar la configuración inicial, que consistirá en: • Determinar la carpeta de almacenamiento		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.25: Requisito RS01

RS02			
Descripción	El sistema debe almacenar la configuración introducida por el usuario en la Base de Datos.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.26: Requisito RS02

RS03			
Descripción	Al inicializar el sistema se debe crear la Base de Datos y crear las etiquetas por defecto: • Extensión • Tamaño • Etiquetas		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.27: Requisito RS03

RS04			
Descripción	El sistema permitirá al usuario crear nuevas etiquetas y sub-etiquetas dentro de la etiqueta por defecto 'ETQ' (Etiquetas)		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.28: Requisito RS04

RS05			
Descripción	El sistema no permitirá al usuario crear etiquetas dentro de las etiquetas por defecto: • 'EXT' (Extensión) • 'TAM' (Tamaño)		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.29: Requisito RS05

RS06			
Descripción	El sistema no permitirá al usuario crear más de dos niveles de etiquetas dentro de 'ETQ' (Etiquetas).		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.30: Requisito RS06

RS07			
Descripción	El sistema creará automáticamente las sub-etiquetas de Extensión y Tamaño (al insertar los archivos)		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.31: Requisito RS07

RS08			
Descripción	Cuando se crean las etiquetas, el sistema debe crear una nueva entrada en la Base de Datos con los siguientes campos: • Nombre de la etiqueta • Etiqueta padre • Permisos (sólo lectura o lectura y escritura) • Nivel de la etiqueta • Número de archivos que contiene		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.32: Requisito RS08

RS09			
Descripción	El sistema debe permitir al usuario eliminar las etiquetas creadas por él.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.33: Requisito RS09

RS10			
Descripción	El sistema no debe permitir al usuario eliminar las etiquetas por defecto (Extensión y Tamaño), ni sus sub-etiquetas		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.34: Requisito RS10

RS11			
Descripción	Cuando se elimina una etiqueta, el sistema debe eliminar la entrada de la Base de Datos correspondiente a dicha etiqueta.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.35: Requisito RS11

RS12			
Descripción	El sistema debe permitir al usuario cambiar una etiqueta de un archivo por otra. Para ello implementará la operación mover un archivo de una etiqueta a otra.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.36: Requisito RS12

RS13			
Descripción	El sistema debe permitir al usuario añadir una etiqueta a un archivo. Para ello implementará la operación copiar un archivo a una nueva etiqueta.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.37: Requisito RS13

RS14			
Descripción	Al añadir o cambiar una etiqueta el sistema debe modificar el campo ‘ETQ’ de la entrada asociada al archivo.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.38: Requisito RS14

RS15			
Descripción	El sistema debe permitir al usuario suprimir la asociación entre una etiqueta y un archivo. Debe permitir eliminar una etiqueta de un archivo.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.39: Requisito RS15

RS16			
Descripción	Al eliminar una etiqueta de un archivo el sistema debe modificar la entrada de la BD correspondiente a dicho archivo.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.40: Requisito RS16

RS17			
Descripción	El sistema no debe permitir al usuario eliminar de un archivo una etiqueta por defecto.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.41: Requisito RS17

RS18			
Descripción	El sistema permitirá al usuario listar las etiquetas disponibles en el sistema. Para ello el usuario deberá indicar al sistema la etiqueta cuyo contenido desea listar o una palabra clave en caso de querer listar todas.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.42: Requisito RS18

RS19			
Descripción	El sistema debe permitir al usuario insertar archivos.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.43: Requisito RS19

RS20			
Descripción	Al insertar un archivo en el sistema, se le asociarán dos etiquetas automáticamente: tamaño y extensión.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.44: Requisito RS20

RS21			
Descripción	Al insertar un archivo, el sistema debe crear una entrada en la Base de Datos asociada a dicho archivo, que contenga los siguientes campos: • Nombre del archivo • Nombre físico • Tamaño • Md5 • Número de etiquetas asociadas al archivo (2 por defecto) • Etiqueta EXT (Extensión) • Etiqueta TAM (Tamaño)		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.45: Requisito RS21

RS22			
Descripción	El sistema debe permitir al usuario cambiar el nombre de un archivo insertado en el sistema.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.46: Requisito RS22

RS23			
Descripción	El sistema debe permitir al usuario añadir una descripción a los archivos insertados en el sistema.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.47: Requisito RS23

RS24			
Descripción	Cuando un usuario cambia el nombre de un archivo o añade a este una descripción, el sistema debe realizar las operaciones de actualización oportunas en la Base de Datos.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.48: Requisito RS24

RS25			
Descripción	El sistema permitirá al usuario realizar búsquedas por el nombre del archivo.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.49: Requisito RS25

RS26			
Descripción	El sistema podrá realizar búsquedas, por el nombre de archivo, mediante patrones o cadenas de caracteres contenidas en el nombre original.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.50: Requisito RS26

RS27			
Descripción	El sistema permitirá al usuario realizar búsquedas por la extensión de los archivos.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.51: Requisito RS27

RS28			
Descripción	El sistema permitirá al usuario buscar archivos más pequeños de un tamaño elegido por este en el momento de realizar la búsqueda.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.52: Requisito RS28

RS29			
Descripción	El sistema permitirá al usuario buscar archivos más grandes de un tamaño elegido por este en el momento de realizar la búsqueda.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.53: Requisito RS29

RS30			
Descripción	El sistema permitirá al usuario buscar archivos en un intervalo de tamaño, elegido por él mismo en el momento de la búsqueda.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.54: Requisito RS30

RS31			
Descripción	El sistema permitirá al usuario realizar búsquedas por etiquetas.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.55: Requisito RS31

RS32			
Descripción	El sistema podrá buscar hasta tres etiquetas en la misma búsqueda.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.56: Requisito RS32

RS33			
Descripción	El sistema debe permitir al usuario realizar búsquedas de archivos por su md5.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.57: Requisito RS33

RS34			
Descripción	El sistema debe permitir al usuario realizar búsquedas combinadas, dando la posibilidad de buscar archivos que cumplan alguna o todas las condiciones de búsqueda por los siguientes campos: • Nombre • Extensión • Tamaño (mayor, menor o intervalo) • Etiquetas (hasta 3 en la misma búsqueda) • Md5		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.58: Requisito RS34

RS35			
Descripción	En las búsquedas, el sistema debe mostrar los siguientes campos de los archivos encontrados: • Nombre • Nombre físico • Extensión • Etiqueta de Tamaño • Etiquetas • Md5		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Media	Verificabilidad	Alta

Tabla 4.59: Requisito RS35

RS36			
Descripción	Al listar las etiquetas o buscar archivos el sistema debe hacer consultas a la BD.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Media

Tabla 4.60: Requisito RS36

RS37			
Descripción	El sistema debe permitir el usuario eliminar archivos del sistema.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.61: Requisito RS37

RS38			
Descripción	Al eliminar un archivo el sistema debe eliminar físicamente el archivo de la carpeta de almacenamiento y eliminar la entrada correspondiente de la Base de datos.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.62: Requisito RS38

RS39			
Descripción	El sistema debe permitir al usuario abrir los archivos insertados en el sistema.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Alta

Tabla 4.63: Requisito RS39

RS40			
Descripción	Cuando el usuario desee abrir un archivo, el sistema debe comprobar (consultando la BD) si existe un programa determinado para abrir la extensión del archivo. S existe, el archivo se abrirá con dicho programa y si no, se abrirá con el programa por defecto.		
Tipo	Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Media

Tabla 4.64: Requisito RS40

RS41			
Descripción	El sistema debe permitir al usuario insertar programas que abran las diferentes extensiones de archivo. Una vez introducidos, el sistema debe almacenar los programas en la configuración del sistema (Base de Datos).		
Tipo	Funcional	Necesidad	Opcional
Prioridad	Baja	Verificabilidad	Alta

Tabla 4.65: Requisito RS41

RS42			
Descripción	El sistema debe ser desarrollado en un lenguaje de programación ágil.		
Tipo	No Funcional	Necesidad	Esencial
Prioridad	Alta	Verificabilidad	Media

Tabla 4.66: Requisito RS3742

Matriz de Trazabilidad: Requisitos de Usuario – Requisitos de Software

RS	RU01	RU02	RU03	RU04	RU05	RU06	RU07	RU08	RU09	RU10	RU11	RU12	RU13	RU14	RU15	RU16	RU17	RU18	RU19	RU20	RU21	RU22	RU23
01	X																						
02	X																						
03	X																						
04		X																					
05			X																				
06				X																			
07		X																					
08		X																					
09					X																		
10						X																	
11					X																		
12							X																
13								X															
14							X	X															
15									X														
16									X														
17										X													
18											X												
19												X											
20												X											
21												X											
22													X										
23														X									
24													X	X									
25															X								
26															X								
27																X							
28																	X						
29																	X						
30																	X						
31																		X					
32																		X					
33																			X				
34																				X			
35															X	X	X	X	X	X			
36											X				X	X	X	X	X	X			
37																					X		
38																					X		
39																						X	
40																						X	
41																							X
42																							

Tabla 4.67: Matriz de Trazabilidad: RU- RS

4.2.3 Base de Datos

Para la gestión de etiquetas y de archivos es necesaria la creación de una Base de Datos. A continuación se muestra el modelo entidad - relación de dicha BD.

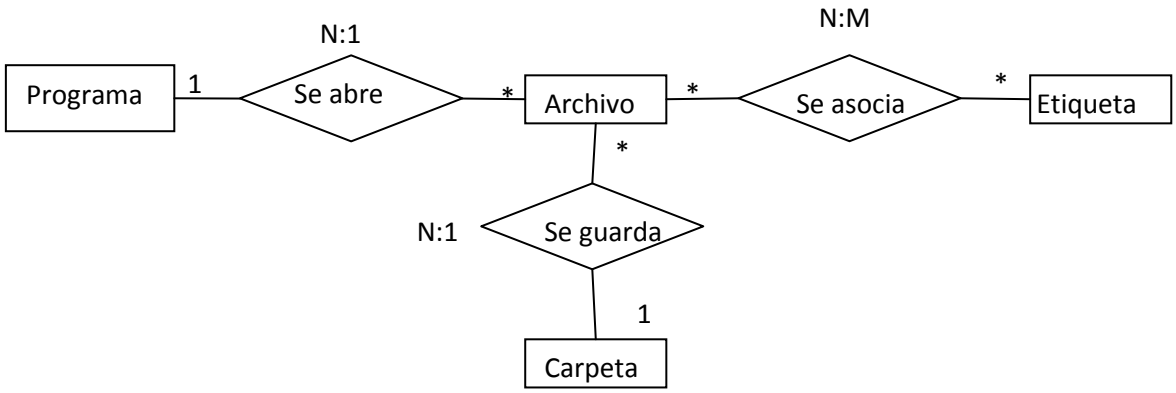

Figura 4.5: Modelo entidad-relación

Tal y como muestra la Figura 4.5, existen 4 tipos de entidad Archivo, Etiqueta, Carpeta y Programa. En las tablas siguientes se definen las propiedades de cada entidad.

Nombre	Archivo
Objeto	Almacenar la información de los archivos insertados en el sistema de ficheros
Alcance	Se entiende como archivo un fichero de datos con extensión.
Clave primaria	Id único. Número hexadecimal de 24 dígitos

Tabla 4.68: Entidad Archivo

Nombre	Etiqueta
Objeto	Almacenar la información de las etiquetas existentes en el sistema
Alcance	Se entiende como etiqueta la taxonomía que ha sido definida en el sistema y permite la localización de los archivos.
Clave primaria	Id único. Número hexadecimal de 24 dígitos

Tabla 4.69: Entidad Etiqueta

Nombre	Carpeta
Objeto	Almacenar la información de la carpeta de almacenamiento del sistema
Alcance	Se entiende como carpeta un directorio del sistema de ficheros
Clave primaria	Id único. Número hexadecimal de 24 dígitos

Tabla 4.70: Entidad Carpeta

Nombre	Programa
Objeto	Almacenar la información de la carpeta de almacenamiento del sistema
Alcance	Se entiende como programa una aplicación que abre las diferentes extensiones de los archivos
Clave primaria	Id único. Número hexadecimal de 24 dígitos

Tabla 4.71: Entidad Programa

4.3 Diseño

En este capítulo se expondrá el diseño del sistema a desarrollar. En primer lugar se expondrán las opciones de diseño y la escogida en cada caso. A continuación se mostrará la arquitectura del sistema y la estructura de la base de datos y, por último, se explicará el diseño del espacio de nombres del sistema.

4.3.1 Opciones de Diseño

Al realizar el estudio y análisis del sistema de ficheros CollectionFS, se contemplaron distintas opciones para realizar el diseño. A continuación se muestra el análisis de esas opciones, así como la opción escogida en cada caso, y las razones de dicha elección.

Lenguaje de Programación

Dado que el sistema a desarrollar tiene que ser sencillo y funcional, es necesario que el lenguaje que lo desarrolle sea ágil. Los lenguajes ágiles contemplados para la realización del presente Trabajo Fin de Grado son Java y Python.

Dados los requisitos del sistema a desarrollar, es necesario escoger un lenguaje con características como:

- Simplicidad: es necesario que el lenguaje adoptado sea sencillo de aprender y de utilizar.
- Integración con FUSE: debido a que parte del sistema estará integrado con FUSE, es necesario que el lenguaje escogido esté, así mismo, integrado con FUSE.
- Alto nivel: para que el desarrollo del sistema sea sencillo y rápido es necesario utilizar un lenguaje de alto nivel.

	Python	Java
Simplicidad	✓	
Integración con FUSE	✓	
Alto nivel	✓	✓

Tabla 4.72: Opciones de Diseño. Lenguaje de Programación

El lenguaje de programación Python cumple con todas las características necesarias para el desarrollo del sistema de ficheros, a diferencia de Java, lo que demuestra que es la mejor opción y por lo tanto la elegida.

Base de Datos

Para el desarrollo del sistema de ficheros semántico aquí expuesto, es necesaria la utilización de una Base de Datos, que contenga las etiquetas del sistema y los metadatos de los ficheros, para hacer posible su organización.

Los requisitos necesarios que tiene que cumplir la Base de Datos utilizada son:

- Rapidez: las inserciones y búsquedas han de ser rápidas. Al ser un sistema de ficheros cuyo objetivo principal es el de facilitar y agilizar la búsqueda de documentos, es importante que la Base de Datos sea lo más rápida posible en las inserciones y búsquedas.
- Sencillez en la Implementación: es necesario que la implementación, tanto de la Base de Datos como de las operaciones sobre ella, sea sencilla, para poder gestionar los datos de una forma fácil y segura.
- Organización: es necesario la flexibilidad a la hora de guardar los datos. Cada archivo almacenado en el sistema de ficheros poseerá unas propiedades diferentes, y podrá ser etiquetado de múltiples maneras, por lo que los datos introducidos en la BD no serán homogéneos. La BD a utilizar deberá permitir esta flexibilidad en la organización.

En la Tabla 4.73 se muestra qué tipo de Base de Datos (*SQL* o *NoSQL*) cumple con los requisitos anteriormente mencionados.

	Base de Datos *SQL*	**Base de Datos *NoSQL***
Rapidez		✓
Sencillez Implementación		✓
Organización		✓

Tabla 4.73: Opciones de Diseño. Base de Datos

Se puede observar que el tipo de Base de Datos *NoSQL* cumple con todos los requisitos necesarios, lo que demuestra que es la mejor opción para el sistema a desarrollar. Sin embargo, dado que ambos tipos de Bases de Datos poseen numerosas implementaciones, es necesario hacer una comparación más profunda entre distintos Sistemas Gestores de Bases de Datos para asegurarnos de tomar la opción más adecuada. En la Tabla 4.74 se realiza una comparación entre una Base de Datos *SQL* (*MySQL*) y otra *NoSQL* (*MongoDB*). Se han escogido

estos dos Sistemas de Bases de Datos debido a su integración con Python (lenguaje escogido para el desarrollo) y por ser *open source*.

	MySQL	MongoDB	Razonamiento
Rapidez		✓	A pesar de que *MySQL* proporciona rapidez en inserciones y búsquedas, *MongoDB* tiene mejor rendimiento en dichas operaciones [22].
Compatibilidad	✓		*MySQL* posee gran compatibilidad entre distintos sistemas o plataformas. *MongoDB*, por ser una implementación más reciente no proporciona tanta compatibilidad.
Transacciones	✓		Uno de los problemas de las BD *NoSQL* (y por lo tanto de *MongoDB*) es que no garantizan las transacciones, mientras que *MySQL* sí las garantiza proporcionando soporte para control de transacciones.
Flexibilidad		✓	*MySQL*, al ser una base de datos relacional, exige una organización de los datos en tablas rígidas. *MongoDB* permite el almacenamiento de campos heterogéneos en la misma colección.
Replicación	✓	✓	Ambos tipos de Bases de Datos proporcionan replicación maestro-esclavo.

Tabla 4.74: Comparación *MySQL* y *MongoDB*

En la comparación realizada en la Tabla 4.74 se puede observar que ambas Bases de Datos están igualadas en prestaciones. Sin embargo es necesario analizar qué es necesario para el presente proyecto.

- La Base de Datos seleccionada debe tener buen rendimiento en las operaciones de inserción y búsqueda, por lo mencionado anteriormente.
- Es necesaria una Base de Datos sin un esquema de almacenamiento rígido, ya que los datos de los ficheros son heterogéneos.
- Todas las operaciones realizadas sobre la Base de Datos serán sencillas, es decir, no será necesaria la implementación de transacciones.
- El sistema de ficheros aquí presentado se analiza y diseña para su implementación en el Sistema Operativo Ubuntu y con el lenguaje de programación Python, por lo que no es necesaria la compatibilidad con otras plataformas.

Dado que *MongoDB* proporciona rapidez en las operaciones y flexibilidad en el almacenamiento, y el sistema no necesita fiabilidad en las transacciones ni compatibilidad con

otras plataformas, es la opción más adecuada para el presente trabajo y por lo tanto la escogida.

4.3.2 Arquitectura Software

El sistema de ficheros a desarrollar se compone de dos partes diferenciadas:

- Operaciones sobre la Base de Datos
- Integración del sistema con FUSE

Las operaciones sobre la Base de Datos se realizarán mediante distintas aplicaciones simples que dotan al sistema de la funcionalidad analizada en los casos de uso y requisitos. Algunas de dichas aplicaciones se integrarán con el sistema de ficheros a través de FUSE. En la Figura 4.6 se muestra la estructura del sistema.

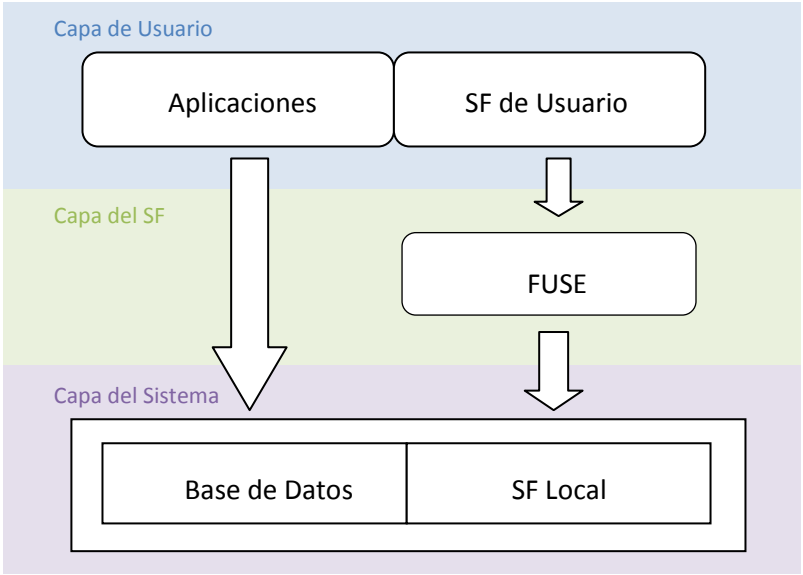

Figura 4.6: Estructura del Sistema

Por lo tanto los componentes que forman el sistema son:

- Aplicaciones: responden a la funcionalidad analizada en los casos de uso y requisitos
- Base de Datos: almacena la gestión de metadatos
- Integración con FUSE: integración de cierta funcionalidad en el sistema de ficheros
- Componente de Almacenamiento: almacena los archivos físicos insertados en el sistema

Las aplicaciones mencionadas responden a la siguiente funcionalidad:

- Inicializar el sistema
- Insertar archivos
- Crear etiquetas
- Añadir etiquetas
- Cambiar una etiqueta por otra
- Eliminar archivos
- Eliminar etiquetas
- Eliminar una etiqueta de un archivo
- Listar las etiquetas y ficheros del sistema
- Encontrar archivos según diferentes parámetros de búsqueda
- Abrir los archivos insertados en el sistema
- Cambiar el nombre y la descripción de un archivo
- Definir los programas que se utilizarán para abrir las diferentes extensiones de archivo

La funcionalidad integrada en el sistema de ficheros a través de FUSE será la de solo lectura.

- Mostrar los directorios y archivos en el navegador del sistema de ficheros
- Abrir ficheros

Se integrará a través de FUSE sólo la funcionalidad de sólo lectura debido a que la interfaz de POSIX no se adapta al sistema de ficheros expuesto en el presente documento. Sería necesario un estudio más riguroso, lo que no es objeto del presente TFG.

4.3.3 Estructura Base de Datos

El sistema de ficheros presentado en el presente TFG utiliza una Base de Datos para la gestión de archivos y etiquetas. Dicha Base de Datos es no relacional y documental, por lo tanto se divide en colecciones y documentos. En el presente apartado se partirá del modelo relacional expuesto en el análisis para diseñar la estructura lógica y física de la Base de Datos utilizada en el sistema a desarrollar.

En la Figura 4.7 se expone el esquema lógico de las relaciones entre documentos.

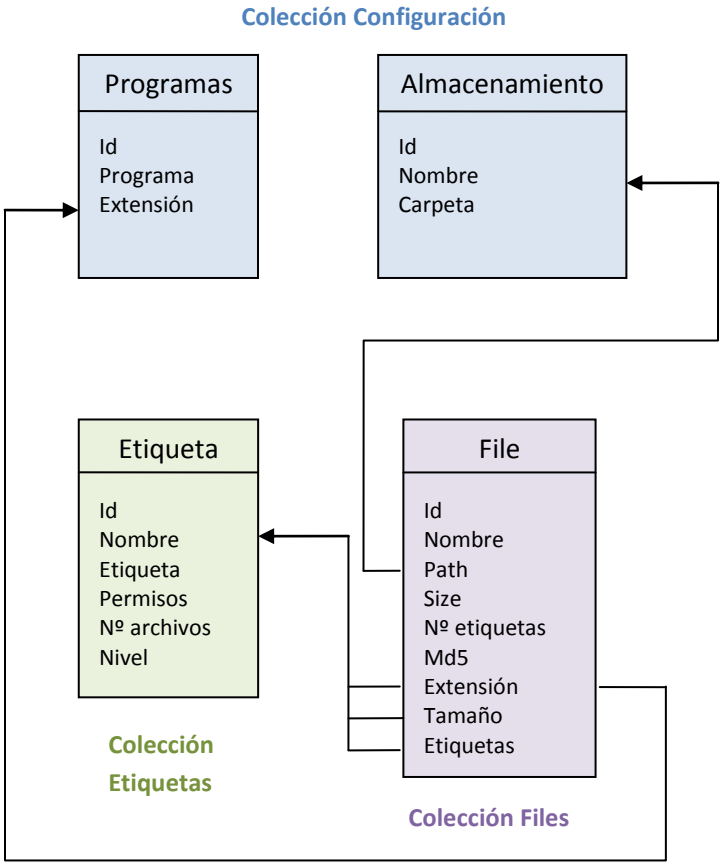

Figura 4.7: Estructura Relacional de la BD

A continuación se especifica qué datos contiene cada campo en los diferentes documentos así como el tipo de datos que se utiliza para almacenarlos.

Colección Configuración

Está formada por dos tipos de documentos diferentes:

- Documento 1: guarda el *path* de la carpeta de almacenamiento del sistema de ficheros

Campo	Tipo de Datos	Descripción
nombre	Cadena de caracteres	Contiene la palabra clave 'folder'. Utilizado para realizar búsquedas.
Folder	Cadena de caracteres	Almacena el path de la carpeta de almacenamiento

Tabla 4.75: Campos del Documento 1 colección Configuración

- Documento 2: guarda los programas que serán utilizados para abrir las diferentes extensiones de archivo.

Campo	Tipo de Datos	Descripción
ext	Cadena de caracteres	Almacena la extensión
prog	Cadena de caracteres	Almacena el programa

Tabla 4.76: Campos del Documento 2 colección Configuración

Colección Etiquetas

Está formada por un tipo de documento. Los documentos de esta colección almacenan las etiquetas del sistema, así como los permisos de estas y etiquetas padre.

Campo	Tipo de Datos	Descripción
nombre	Cadena de caracteres	Almacena el nombre de la etiqueta
etq	Cadena de caracteres	Almacena la etiqueta padre o '' en caso de las etiquetas por defecto.
permisos	Cadena de caracteres	Almacena los permisos de la etiqueta: • r: sólo lectura • rw: lectura y escritura
numFiles	Integer	Almacena el número de archivos que contiene la etiqueta
nivel	Integer	Almacena el nivel de la etiqueta en la jerarquía

Tabla 4.77: Campos de los documentos de la colección Etiquetas

Colección Files

Está formada por un solo tipo de documento. Los documentos de esta colección almacenan la información de los archivos que contiene el sistema. A continuación se expone qué campos forman dichos documentos y el tipo de datos e información de cada uno.

Campo	Tipo de Datos	Descripción
nombre	Cadena de caracteres	Almacena el nombre lógico del fichero
path	Cadena de caracteres	Almacena el nombre físico del archivo (path en la carpeta de almacenamiento)
size	NumberLong	Almacena el tamaño exacto del archivo
nEtq	Integer	Almacena el número de etiquetas asociadas al archivo
md5	Cadena de caracteres	Almacena el MD5 del fichero
EXT	Cadena de caracteres	Almacena la etiqueta extensión del archivo
TAM	Cadena de caracteres	Almacena la etiqueta tamaño del archivo
ETQ	Array de diccionarios	Almacena las diferentes etiquetas definidas por el usuario

Tabla 4.78: Campos de los documentos de la colección Files

4.3.3 Espacio de Nombres

CollectionFS es un sistema de ficheros semántico. Tal y como se ha mencionado en el punto 2.2, "los sistemas de ficheros semánticos son sistemas de ficheros utilizados para la persistencia de la información, estructurando los datos de acuerdo a su semántica y propósito. Permiten ordenar y recuperar los datos por su contenido". En el caso de CollectionFS, los datos se estructuran por medio de etiquetas.

Definición de etiqueta

En el sistema de ficheros aquí tratado, se ha considerado como etiqueta al componente más simple de la taxonomía del sistema. CollectionFS se compone de una jerarquía de etiquetas (o taxones) que permiten la clasificación de los diferentes archivos insertados.

Dicha jerarquía se puede dividir en dos:

- Etiquetas definidas por el sistema
- Etiquetas gestionadas por el usuario.

Las etiquetas definidas por el sistema son aquellas que el sistema asigna automáticamente a los archivos insertados, y el usuario no tiene permiso para modificarlas. Dichas etiquetas son Extensión y Tamaño. Dado que el sistema de ficheros aquí tratado es un sistema de clasificación de la información, se ha determinado que dicha información sea de consulta, de sólo lectura, de tal forma que propiedades como la extensión o el tamaño de un archivo sean permanentes. En la Figura 4.8 se muestra la jerarquía de estas etiquetas definidas por el sistema.

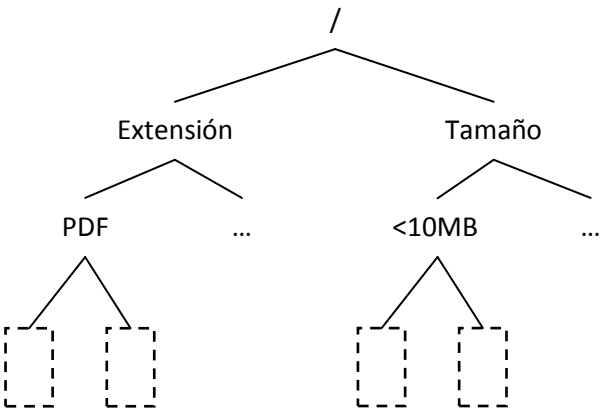

Figura 4.8: Jerarquía Etiquetas Sistema

Las etiquetas gestionadas por el usuario, son taxones de la jerarquía que el usuario crea, modifica y elimina según las necesidades que le vayan surgiendo. En la jerarquía del sistema, existe una etiqueta por defecto llamada 'Etiquetas' que será el raíz de todas las etiquetas creadas por el usuario. Dentro de dicha etiqueta podrán existir hasta 2 niveles de etiquetas, pudiendo asignar a los archivos tanto las etiquetas de primer nivel como las del segundo nivel. La Figura 4.9 contiene esta jerarquía de forma gráfica.

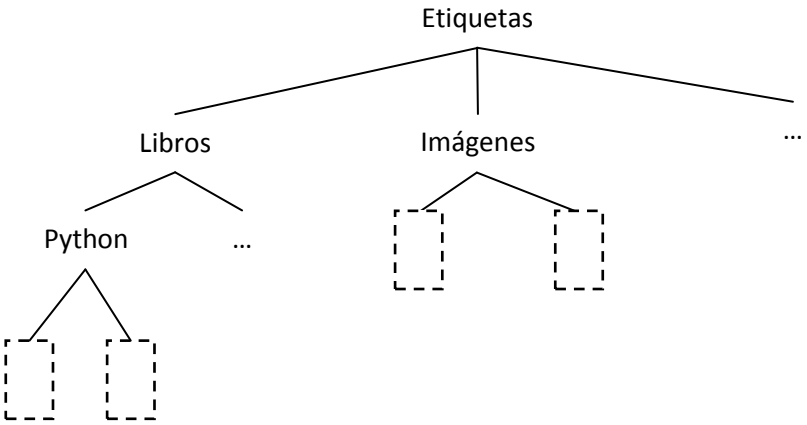

Figura 4.9: Jerarquía Etiquetas de Usuario

Cuando un archivo es insertado, se le asocian las etiquetas definidas por el propio sistema (Extensión y Tamaño), y el usuario puede añadir las etiquetas que considere adecuadas de la colección que él mismo ha creado. El número de etiquetas a asociar a un archivo es, por tanto, como mínimo dos y como máximo todas las etiquetas creadas por el usuario más dos. En la Figura 4.10 se muestra la jerarquía de etiquetas completa del sistema.

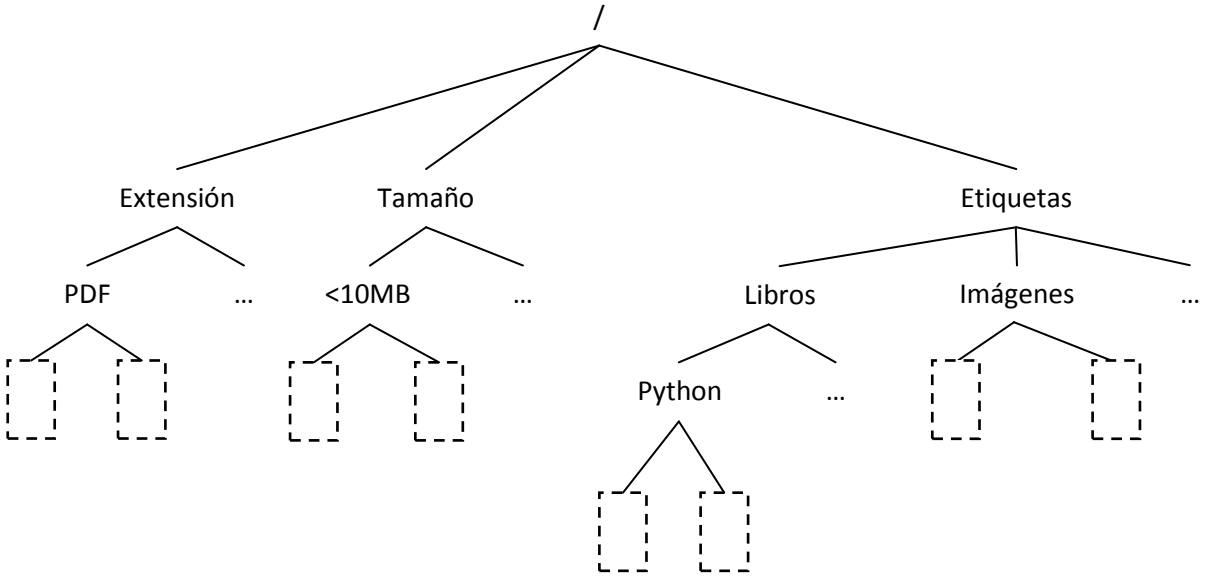

Figura 4.10: Jerarquía Sistema

Se puede observar que la jerarquía es en árbol, es decir, 1 padre N hijos. Se consideró diseñar una jerarquía de N padres M hijos, pero finalmente se decidió simplificarlo a la jerarquía mostrada, por los siguientes motivos:

- La gestión de los archivos sería mucho más complicada
- La localización de un fichero sería más compleja y llevaría más tiempo
- No aportaría prácticamente ninguna mejora en la gestión y localización de archivos

De esta forma, un mismo archivo puede ser localizado de múltiples formas, tantas como etiquetas tenga asociadas, lo que facilita enormemente la recuperación de la información. Además, el nombre de un archivo se puede repetir sin que sea un problema a la hora de localizarlo, ya que en el sistema un archivo se identifica por un id único, y se almacena información propia de él como el MD5 o el tamaño exacto. Esto facilita la gestión de nombres al usuario, ya que puede nombrar los archivos de forma natural, sin tener que inventarse nombres únicos y difíciles de recordar para no sobrescribir ficheros.

5.Desarrollo

Este capítulo está dedicado a los detalles de implementación del prototipo desarrollado. Las versiones utilizadas para el desarrollo son las mostradas en la Tabla 5.1.

MongoDB	2.7.2
Python	2.0.2
FUSE	2.8.4

Tabla 5.1: Versionado

5.1 Aplicaciones

A continuación se exponen los algoritmos utilizados (en pseudocódigo) para implementar las aplicaciones utilizadas para el manejo del sistema de ficheros (4.3.2 Arquitectura Software).

5.1.1 Inicializar Sistema

La aplicación *CFSinit* es la encargada de inicializar el sistema de ficheros. Para ello crea la Base de Datos, establece el directorio que servirá como almacenamiento de los ficheros insertados, guarda la configuración del sistema y crea las etiquetas iniciales.

```
CFSinit(path):
   • Iniciar Base de Datos
   • Insertar path en la BD
   • Crear etiquetas iniciales: EXT, TAM y ETQ
   • Otorgar premisos de lectura 'r' a las etiquetas
```

5.1.2 Insertar Archivo

La aplicación *CFSinsert* es la encargada de insertar los archivos en el sistema de ficheros. Para ello comprueba el tamaño y la extensión del archivo insertado y calcula su md5. A continuación copia el archivo en la carpeta de almacenamiento, renombrándolo con el identificador que lo asocia a la entrada de la base de datos correspondiente. En dicha entrada se guardan los metadatos del fichero (nombre, nombre físico, tamaño, md5, número de etiquetas asociadas) y las etiquetas de extensión y tamaño.

CFSinsert(archivo):

- Calcular tamaño de archivo (*size*)
- Definir la etiqueta Tamaño, con el tamaño calculado (*tam*)
- Comprobar la extensión del archivo (*ext*)
- Calcular el MD5 del fichero (*md5*)

- Buscar en la Base de Datos un archivo con el md5 y el tamaño calculados
- Si existe:
  - Se imprime mensaje de error
- Si no:
  - Insertar los metadatos del fichero en la BD (*nombre*, *size*, *tam*, *ext*, *md5*)
  - Aumentar el contador de archivos en las etiquetas en las que se ha insertado el archivo
  - (si no existen las etiquetas se crean)
  - Copiar archivo en la carpeta de almacenamiento
  - Añadir a la entrada de la BD el path del archivo insertado

5.1.3 Añadir Etiqueta

La aplicación *CFScopy* cumple con la funcionalidad de añadir una etiqueta del sistema a un archivo. En primer lugar comprueba si la etiqueta a añadir existe en el sistema. Si no es así devuelve un mensaje de error. Si la etiqueta existe, comprueba sus permisos. Si los permisos son de lectura y escritura, añade la etiqueta al archivo y aumenta el contador de archivos de dicha etiqueta. Si los permisos son de solo lectura devuelve un mensaje de error.

CFScopy(id, etiqueta):

- Buscar en la BD si existe la etiqueta *etiqueta*
- Si existe:
  - Almacenar los permisos de la etiqueta en *permisos*
  - Almacenar la etiqueta padre en *etq*
- Si no:
  - Se imprime mensaje de error
  - Terminar
- Si permisos = lectura y escritura:
  - Actualizar la entrada de la BD con la nueva etiqueta.
  - Incrementar el número de etiquetas asociadas al archivo
  - Incrementar el número de archivos que contiene la etiqueta
- Si no:
  - Se imprime mensaje de error

5.1.4 Cambiar Etiqueta

La aplicación *CFSmove* es la encargada de cambiar una etiqueta por otra. Esta operación implica las operaciones 'añadir etiqueta' (*CFScopy*) y 'borrar etiqueta de un archivo' (*CFSrmTag*), por lo tanto la aplicación *CFSmove* simplemente llama a esas dos aplicaciones.

```
CFSmove(idd, from, to):
   • llamadaSistema('CFScopy '+ id + ' ' + to)
   • llamadaSistema('CFSrmTag' + id + ' ' + from)
```

5.1.5 Abrir Archivos

Con la aplicación *CFSopen* se pueden abrir los archivos contenidos en el sistema de ficheros. Recibe como parámetro el nombre físico del archivo (path del archivo en la carpeta de almacenamiento). En primer lugar comprueba (con una consulta a la Base de Datos) la extensión real del archivo a abrir. A continuación comprueba en la configuración del sistema con qué programa se abre dicha extensión y abre el archivo con el programa encontrado.

```
CFSopen(id):
   • Buscar la extensión del archivo (id) en la BD
   • Buscar en la BD si existe un programa para dicha
     extensión
   • Si existe:
        o Se abre el archivo con el programa encontrado
   • Si no existe:
        o Se abre el archivo con el programa por defecto
```

5.1.6 Crear Etiqueta

La aplicación *CFSmkdir* es la encargada de crear nuevas etiquetas definidas por el usuario. Recibe como parámetros la etiqueta padre en la que se desea crear y el nombre de la nueva etiqueta. En primer lugar se comprueba que no exista la etiqueta que el usuario intenta crear. Si existe se devuelve un mensaje de error, ya que para simplificar la implementación y las búsquedas no se permite crear dos etiquetas con el mismo nombre .Si no existe se comprueba si la etiqueta padre tiene los permisos adecuados. Si es así se comprueba el nivel de la etiqueta padre. Si dicho nivel es 0 o 1 se crea la etiqueta, añadiendo una entrada en la Base de Datos.

```
CFSmkdir(padre, nombreEtiqueta):
  • Si existe en la BD una etiqueta igual a nombreEtiqueta:
      o Imprimir mensaje de error
      o Terminar
  • Si la etiqueta padre es EXT o TAM:
      o Imprimir mensaje de error
  • Si no:
      o Si el nivel de la etiqueta padre es 0 o 1:
          ▪ Insertar la etiqueta con permisos lectura y escritura y número de archivos 0
      o Si no:
          ▪ Imprimir mensaje de error
```

5.1.7 Eliminar Archivos

La aplicación *CFSremove* cumple con la funcionalidad de eliminar archivos del sistema. Para ello borra la entrada de la base de datos asociada con el archivo, elimina el archivo físico de la carpeta de almacenamiento y realiza un decremento en el contador de archivos de las etiquetas asociadas al archivo a eliminar.

```
CFSremove(id):
  • Restar 1 al contador de archivos de las etiquetas asociadas al archivo
  • Eliminar la entrada de la BD
  • Eliminar el archivo físico
```

5.1.8 Cambiar Nombre y Descripción

La aplicación *CFSchange* se encarga de dar la posibilidad al usuario de cambiar el nombre de un archivo así como de añadir una descripción. El usuario introduce el nombre y la descripción (o sólo una de ellas) y la aplicación utiliza el campo correspondiente en la base de datos.

```
CFSchange(name, id):
  • Actualizar el nombre del archivo (id) en la BD
  • Pedir al usuario que escriba la descripción
  • Actualizar la descripción en la BD
```

5.1.9 Definir Programas

Esta aplicación se encarga de definir en la base de datos los programas con los que se abrirán las diferentes extensiones de archivo. Para ello se permite al usuario introducir las extensiones y los programas, actualizando en la base de datos si ya existía una entrada para la extensión introducida o insertando una nueva entrada en caso contrario.

```
CFSprograma():
  • a = 0
  • Mientras a == 0:
      o Preguntar al usuario por la extensión a añadir
      o Preguntar al usuario por el programa a añadir
      o Si la extensión introducida existe:
          ▪ Actualizar la BD con el programa
            introducido
      o Si no:
          ▪ Insertar en la BD la extensión y el
            programa introducidos
      o Preguntar al usuario si quiere continuar
      o Si respuesta es sí:
          ▪ a = 1
```

5.1.10 Eliminar Etiquetas

La aplicación *CFSrmdir* es la encargada de eliminar las etiquetas creadas por el usuario. En primer lugar comprueba si la etiqueta que el usuario quiere eliminar existe y qué permisos tiene. Si la etiqueta no existe o no tiene permisos de lectura y escritura, imprime un mensaje de error. Si por el contrario tiene los permisos adecuados se comprueba si la etiqueta tiene archivos asociados. Si es así se pregunta al usuario si está seguro de eliminar la etiqueta y todas sus asociaciones a archivos. Si el usuario acepta, se elimina la etiqueta del sistema y todas las asociaciones. Si la etiqueta no contiene archivos se comprueba si contiene otras etiquetas. Se repite el proceso de preguntar al usuario y si da permiso, se elimina la etiqueta y todas sus sub-etiquetas. En caso de que la etiqueta a eliminar no contenga archivos ni sub-etiquetas, se elimina directamente.

```
CFSrmdir(etiqueta):
   • Comprobar en la BD los permisos de etiqueta
   • Si la etiqueta no existe:
        o Imprimir mensaje de error
   • Si los permisos son sólo lectura:
        o Imprimir mensaje de error
   • Si no:
        o Si la etiqueta contiene archivos:
             ▪ Preguntar al usuario
             ▪ Si acepta:
                  • Buscar en la BD los archivos
                    asociados
                  • Eliminar las asociaciones
                  • Eliminar la etiqueta
        o Si la etiqueta contiene sub-etiquetas:
             ▪ Preguntar al usuario
             ▪ Si acepta:
                  • Buscar en la BD las sub-etiquetas
                  • Buscar en la BD si existen archivos
                    asociados a las sub-etiquetas
                  • Eliminar los archivos
                  • Eliminar las sub-etiquetas
                       o Eliminar la etiqueta
```

5.1.11 Listar Etiquetas y Archivos

La aplicación *CFSls* cumple con la funcionalidad de listar las etiquetas y archivos disponibles en el sistema. Se pueden realizar 6 tipos de listados:

- Lista de todos los archivos
- Lista de todas las etiquetas
- Lista de todas las etiquetas y los archivos que contienen
- Lista de las sub-etiquetas de una etiqueta en concreto
- Lista de los archivos que contiene una etiqueta en concreto
- Lista de los archivos y etiquetas que contiene una etiqueta en concreto

```
CFSls(etiqueta, opciones):
  • Si no se introduce etiqueta ni opciones:
      o Imprimir todas las etiquetas del sistema
  • Si no se ha introducido etiqueta:
      o Si opciones == a:
          ▪ Imprimir todas las etiquetas del sistema con
            su contenido
      o Si opciones == e:
          ▪ Imprimir todas las etiquetas del sistema
      o Si opciones == f:
          ▪ Imprimir todos los archivos insertados en el
            sistema
  • Si se ha introducido etiqueta:
      o Si opciones == a:
          ▪ Imprimir todas las sub-etiquetas de etiqueta
            y todo su contenido
      o Si opciones == e:
          ▪ Imprimir todas las sub-etiquetas de etiqueta
      o Si opciones == f:
          ▪ Imprimir los archivos que contenga etiqueta
```

5.1.12 Eliminar Etiqueta de Archivo

La aplicación *CFSrmTag* cumple con la funcionalidad de eliminar una etiqueta de un archivo. Son posibles dos tipos de borrado:

- Eliminar todas las etiquetas de un archivo (excepto las etiquetas por defecto)
- Eliminar una etiqueta de un archivo

Si el usuario escoge la opción de eliminar todas las etiquetas del archivo, la aplicación busca en la Base de Datos dichas etiquetas y las elimina. Si, por el contrario, escoge la opción de eliminar una etiqueta introducida, se elimina la etiqueta, se resta uno al número de etiquetas del archivo y se resta uno al número de archivos de la etiqueta.

```
CFSrmTag(id, opciones):
  • Si opciones == a:
      o Buscar en la BD las etiquetas asociadas al archivo
      o Restar 1 al número de archivos de las etiquetas
      o Borrar todas las etiquetas asociadas al archivo id
      o Igualar a 2 el número de etiquetas del archivo
        (etiquetas por defecto)
  • Si opciones == t:
      o Comprobar los permisos de la etiqueta introducida
      o Si permisos == sólo lectura:
          ▪ Imprimir mensaje de error
      o Si no:
          ▪ Eliminar la etiqueta del archivo id
          ▪ Restar 1 al número de etiquetas del archivo
          ▪ Restar 1 al número de archivos de la
            etiqueta
```

5.1.13 Buscar

La aplicación *CFSfind* es la encargada de realizar las búsquedas de archivos en el sistema. Recibe como parámetros los diferentes campos por los que el usuario desea buscar. Los campos por los que se pueden realizar búsquedas son:

- Nombre (incluye búsquedas no exactas)
- Extensión
- Etiquetas (hasta 3 etiquetas en la misma búsqueda)
- Md5
- Tamaño
    - Más grande que
    - Más pequeño que
- Descripción (incluye búsquedas no exactas)

Si la aplicación no recibe ningún parámetro muestra todos los ficheros del sistema. Si por el contrario recibe uno o más parámetros de búsqueda, confecciona una lista con los campos de la Base de Datos en los que tiene que buscar y sus respectivos valores. A continuación realiza una búsqueda con todos los elementos de dicha lista y el operador 'and', de tal forma que el resultado de la búsqueda serán los archivos que cumplan todas las condiciones que el usuario ha indicado. Finalmente imprime los resultados de la búsqueda.

Al imprimir los resultados, la aplicación imprime, por cada fichero, los siguientes campos:

- Nombre
- Id
- Extensión
- Tamaño
- Nombre físico
- MD5
- Etiquetas
- Descripción

```
CFSfind(parámetros de búsqueda):
  • Si no hay parámetros de búsqueda:
      o Buscar todos los ficheros del sistema
      o Imprimir resultados
  • Si no:
      o Comprobar parámetros de búsqueda
      o Mientras haya elementos:
          ▪ Buscar elementos que cumplan con todos
            los parámetros de búsqueda (and)
          ▪ Guardar resultados en lista
      o Recorrer lista
          ▪ Eliminar resultados repetidos
      o Imprimir resultados almacenados en lista
```

5.2 Integración con FUSE

Este apartado contiene los algoritmos en pseudocódigo de las operaciones implementadas en FUSE. Para integrar la funcionalidad de sólo lectura (mencionada en el capítulo 4.3.2 Arquitectura Software) a través de FUSE, es necesario implementar las operaciones *getattr*, *readdir* y *read*.

Para la implementación de dichas funciones se han contemplado dos modos:

- Realizar una precarga de todos los directorios y archivos del sistema de ficheros en una lista, de tal forma que no se mostrarán las modificaciones (añadir o eliminar etiquetas y ficheros) realizadas después del montaje.
- Consultar a la Base de Datos cada vez que se realiza una llamada a las diferentes funciones, permitiendo al usuario la modificación del sistema (añadir o eliminar etiquetas y ficheros) después del montaje del mismo.

El espacio de nombres se gestiona a través del id del elemento concatenado al nombre del fichero o la etiqueta, conociendo así, inequívocamente, a qué etiqueta o fichero se está accediendo.

5.2.1 Getattrr

Getattr es la primera llamada que hace FUSE y se invoca para obtener los atributos de un directorio o fichero específico.

```
Getattr(path):
   • Crear una estructura stat en la que la información
     será almacenada
   • Rellenar la estructura con la información adecuada
     (fechas de creación, modificación y acceso, modo,
     id de usuario, etc.)
   • Si el path es un directorio:
        o Dar permisos de directorio
        o Igualar el número de contador de enlaces a 2
   • Si es un fichero
        o Dar permisos de fichero
        o Igualar el número de contador de enlaces a 1
   • Devolver la estructura
```

5.2.2 Readdir

Función invocada cuando se abre un directorio o se lista su contenido.

```
Readdir(path, offset):
   • Rellenar la entrada de fuse con '.' y '..'
   • Analizar path y obtener el directorio a leer
   • Comprobar el contenido del directorio
      o Analizar el nombre y obtener el id del
        directorio a leer
      o Consultar en la BD/Lista precargada los
        elementos que tienen como padre id
   • Rellenar la entrada de fuse con los elementos
     encontrados
```

5.2.4 Read

Esta función se invoca cuando el usuario quiere abrir un fichero del sistema.

```
Read(path, length, offset):
   • Analizar path y obtener el elemento a abrir
   • Comprobar en la BD /Lista precargada si el elemento
     es un fichero
   • Si es un fichero:
      o Abrir fichero
      o Leer length bytes del fichero
      o Cerrar fichero
   • Devolver bytes leídos
```

Una vez realizado el montaje del sistema de ficheros a través de FUSE, es posible la navegación por las distintas etiquetas. En la Figura 5.1 se muestra la estructura de etiquetas y archivos de CollectionFS. Se puede observar que el documento *FUSE.pdf* tiene como etiqueta de extensión *pdf* y como etiquetas añadidas por el usuario *Libros: Fuse*, y es posible acceder a él a través de ambas etiquetas.

En dicha figura se puede observar, así mismo, la gestión de nombres anteriormente mencionada. El nombre de cada elemento tiene la estructura *id – nombre*.

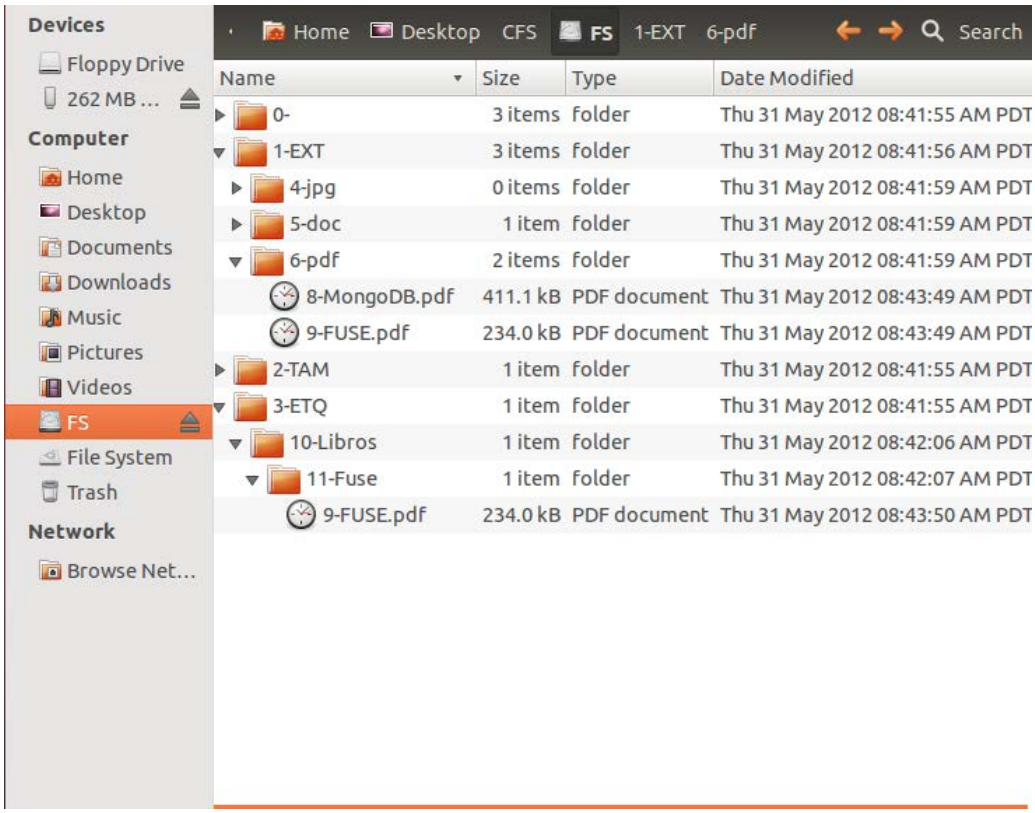

Figura 5.1: Directorios CollectionFS

5.3 Bibliotecas externas

5.3.1 Optparse

Para permitir al usuario introducir distintas opciones en la línea de comandos, se ha utilizado la biblioteca *optparse* [23]. Dicha biblioteca funciona de la siguiente manera: el desarrollador crea una instancia de *OptionParser*, la rellena con opciones y analiza la entrada por línea de comandos. *Optparse* permite a los usuarios especificar opciones con la sintaxis GNU/POSIX convencional, y adicionalmente genera el uso y los mensajes de ayuda automáticamente.

Optparse fue diseñada explícitamente para fomentar la creación de programas sencillos con interfaces de línea de comandos convencionales. Para ello, soporta exclusivamente la sintaxis de línea de comandos más común y la semántica convencionalmente utilizada en Unix.

La terminología utilizada es la siguiente:

- Argumento: cadena de caracteres introducida por la línea de comandos

```
<script> ARG
```

- Opción: argumento utilizado para proporcionar información extra para guiar o personalizar la ejecución de un programa.

```
<script> -h
```

- Opción con argumento: argumento que sigue a una opción, está asociado con dicha opción y es utilizado desde la lista de argumentos cuando la opción está.

```
<script> -n ARG
```

- Opción requerida: opción que debe ser introducida en la línea de comandos

```
<script> (…) -a
```

El mensaje de ayuda que muestra *optparse* tiene el formato siguiente:

```
Usage: <script> [options]

Options:
  -h, --help            show this help message and exit
  -f FILE, --file=FILE  write report to FILE
  -q, --quiet           don't print status messages
```

5.3.2 PyMongo

La distribución PyMongo [24, 25] contiene tres paquetes para interactuar con MongoDB desde Python. El paquete *bson* es una implementación del formato BSON para Python, el paquete *pymongo* es un driver nativo de Python para MongoDB, y *gridfs* es una serie de herramientas para trabajar con la especificación de almacenamiento *GridFS*.

Dicha distribución es soportada en la versión de Python 2.4 o mayor y en la 3.1 o mayor.

5.3.2 Gnome-open

Comando que permite ejecutar desde la terminal varios tipos de acciones [26]. Se puede usar para:

- Abrir páginas de internet. Si se ejecuta `gnome-open http://www.google.es` se abrirá la página en el navegador por defecto.
- Abrir documentos o archivos: para abrir cualquier fichero se debe ejecutar `gnome-open fichero.ext` y se abrirá el fichero con el programa con el que este asociado.
- Abrir directorios: si se quiere abrir el directorio `/home/usr/Documentos`, simplemente habrá que ejecutar `gnome-open /home/usr/Documentos`, y se abrirá dicho directorio en *nautilus*.
- Enviar un correo: si se ejecuta `gnome-open mailito:nombre@gmail.com`, se abrirá el programa de correo predeterminado del sistema en la ventana de redacción de e-mail para escribir un mensaje a la dirección nombre@gmail.com

En el sistema desarrollado en el presente TFG se ha utilizado este comando para abrir los archivos insertados en el sistema.

6. Análisis de Rendimiento

En este capítulo se mostrará la evaluación del sistema diseñado. Para ello se realizarán diferentes pruebas y se analizarán los resultados.

El entorno utilizado para la realización de las pruebas ha sido el mostrado en la Tabla 6.1.

Procesador	Intel(R) Core™ i7-2600 CPU, 3.4 Ghz
Sockets	1
Cores por socket	4
Hyperthreading	Sí
Memoria	8GB
Disco Duro	Segate modelo Barracuda. 1 TeraByte. 7200 revoluciones

Tabla 6.1: Entorno de Pruebas

Las operaciones que se realizarán para medir el rendimiento del sistema son:

- Inserción
- Eliminación
- Búsqueda

A continuación se expone cada prueba mostrando los resultados obtenidos y el análisis correspondiente a dichos resultados.

6.1 Inserción

Para evaluar el rendimiento de la operación insertar, se han realizado pruebas con diferente número de inserciones. El número de archivos insertado ha sido: 5, 10, 50, 100, 500, 1000 y 5000.

Nº Archivos	Tiempo Total (s)	Tiempo Unitario (s)
5	0.75	0.15
10	1.46	0.146
50	7.46	0.149
100	16.28	0.163
500	103.13	0.206
1000	209.84	0.209
5000	1349.95	0.269

Tabla 6.2: Pruebas Inserción

Como se puede observar en la Tabla 5.2, cuantos más archivos se insertan en el sistema, más largo es el tiempo de insertar uno solo. En la Figura 6.1 se puede apreciar mejor este crecimiento. Así mismo, se aprecia que la mayor parte del tiempo en insertar un elemento, se invierte en el cálculo del md5. Las otras operaciones indicadas en el gráfico son comprobaciones de tamaño y extensión, inserciones y consultas a la Base de Datos, etc.

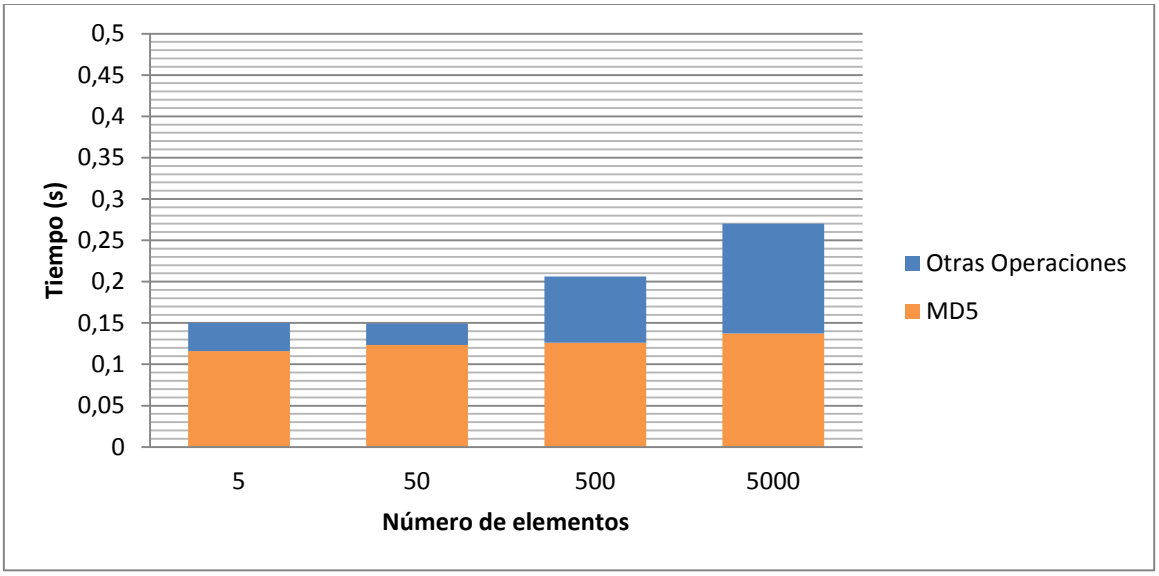

Figura 6.1: Gráfico de tiempo unitario en inserción

La operación más costosa realizada en la inserción es, como se ha mostrado en la Figura 6.1, el cálculo del MD5 de cada archivo. Para comprobar el efecto que tiene dicha operación en el tiempo total de inserción, se ha medido el tiempo que se tarda en calcular el MD5 con diferente número de elementos insertados. La Figura 6.2 muestra el tiempo total en insertar diferente número de archivos y de dicho tiempo, cuánto se ha invertido en el cálculo del MD5. Se puede observar que gran parte del tiempo total de inserción se debe al cálculo del MD5, lo que provoca que la operación Insertar no escale.

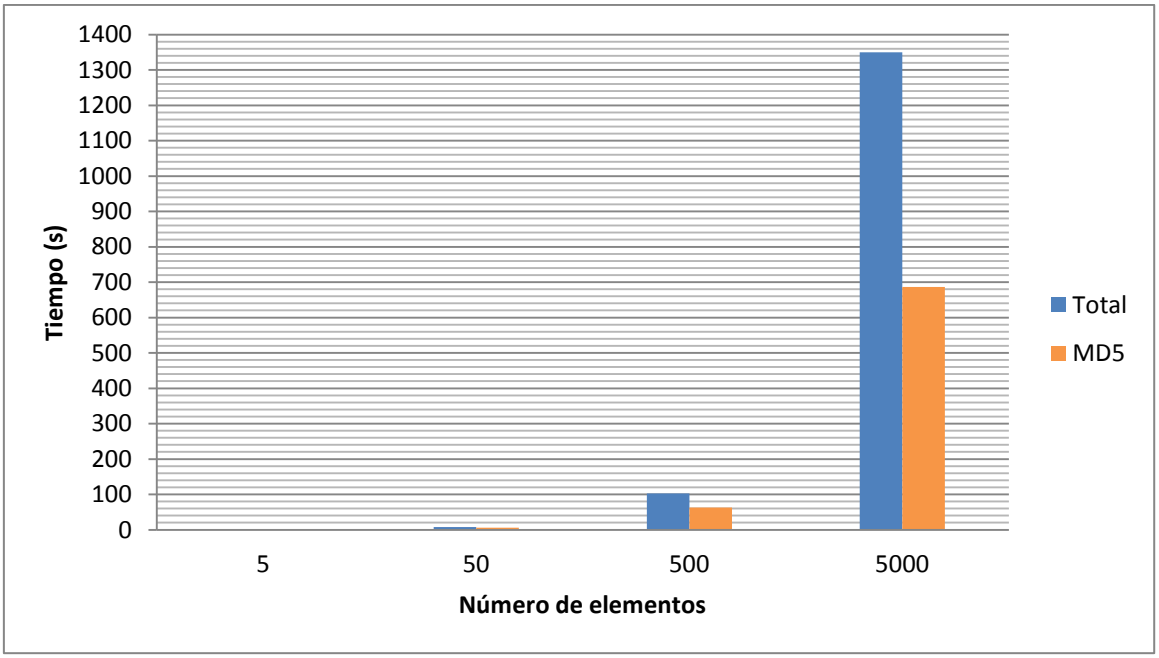

Figura 6.2: Gráfico de tiempo total y tiempo en calcular MD5 Inserción

El mal rendimiento de la operación insertar se debe a el cálculo del md5 por cada archivo insertado en el sistema. Dicho cálculo es costoso ya que para realizar el md5 es necesario leer el archivo completo, y en el caso de inserción de ficheros de gran tamaño, el tiempo aumenta en gran medida.

6.2 Eliminación

Para evaluar el rendimiento de la operación borrar, se han realizado pruebas con diferente número de elementos en el sistema a eliminar. El número de archivos borrado ha sido: 5, 10, 50, 100, 500, 1000 y 5000.

Nº Archivos	Tiempo Total(s)	Tiempo Unitario (s)
5	0.62	0.124
10	1.26	0.126
50	6.21	0.124
100	12.55	0.125
500	61.50	0.123
1000	130.6	0.130
5000	663.65	0.132

Tabla 6.3: Pruebas Eliminación

Tal y como se observa en la Tabla 6.3, la operación de eliminar archivos escala, ya que aún creciendo exponencialmente el número de archivos a borrar, el tiempo en eliminar un único archivo es prácticamente igual en todos los casos. En la Figura 6.3 se pueden observar mejor estos resultados, y cómo la operación borrar escala perfectamente.

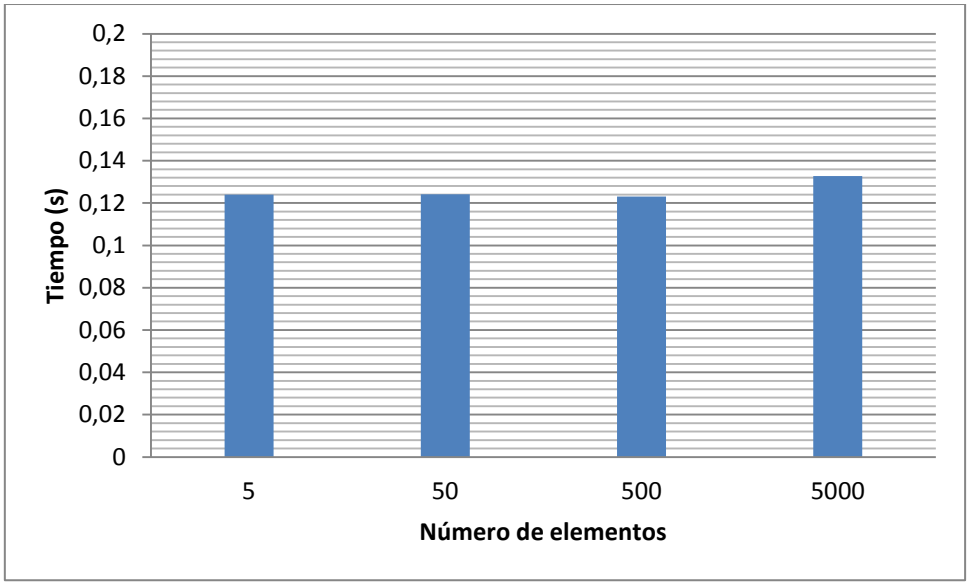

Figura 6.3: Gráfico tiempo unitario eliminación

En la Figura 6.4 se muestra el tiempo total en insertar diferente número de elementos. Se puede observar que el tiempo en eliminar crece exponencialmente debido a que el número de archivos a eliminar crece de dicha forma.

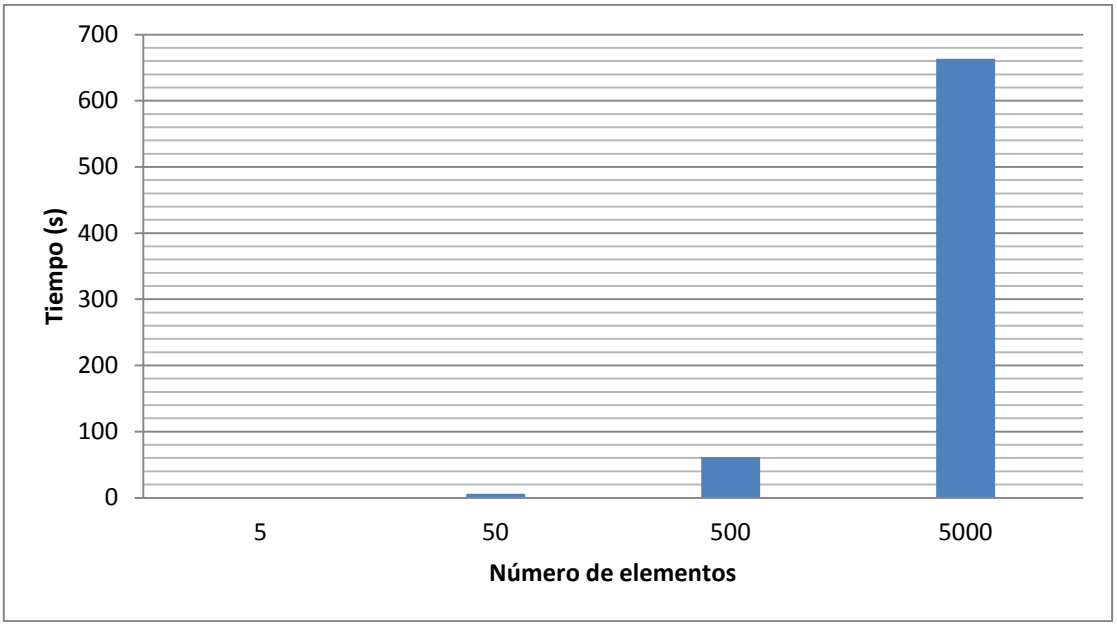

Figura 6.4: Gráfico tiempo total eliminación

La operación eliminar escala perfectamente, siendo el tiempo unitario el mismo al eliminar uno o numerosos archivos. Esto indica que dicha operación tiene un buen rendimiento.

6.3 Búsqueda

Para evaluar el rendimiento de la operación buscar, se han realizado pruebas con diferente número de archivos a encontrar, así como diferentes tipos de búsquedas. En la Tabla 6.4 se muestran los resultados de dichas pruebas, mostrando el tiempo total de cada búsqueda así como el tiempo en encontrar un archivo.

Nº Archivos	Tiempo Total(s)	Tiempo Unitario(s)
1	0.129	0.129
24	0.127	0.0053
58	0.155	0.0026
91	0.159	0.0017
443	0.502	0.0011
858	0.896	0.0010
1690	1.575	0.0009
5016	3.956	0.0008

Tabla 6.4: Pruebas Búsqueda

Como se puede observar en los resultados de las pruebas, la operación buscar tiene muy buen rendimiento, ya que cuantos más elementos se buscan, menos se tarda en encontrar cada uno de ellos. En la Figura 6.5 se muestra gráficamente dicho resultado. Se aprecia la gran diferencia entre el tiempo unitario en la búsqueda de un único elemento y en la búsqueda de gran número de elementos.

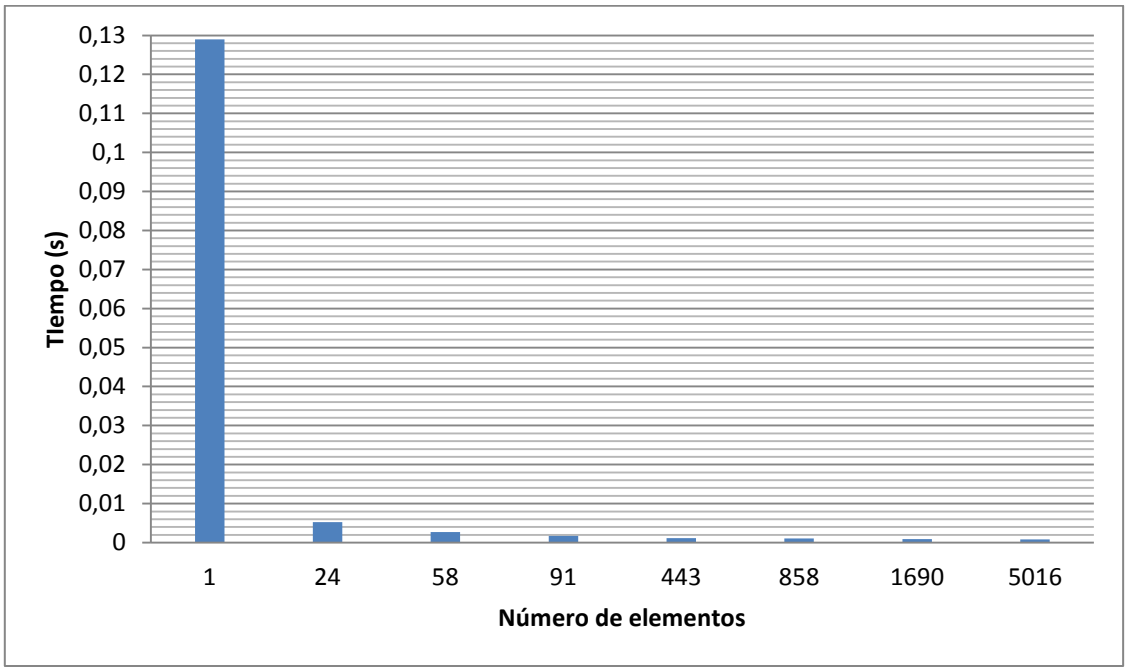

Figura 6.5: Gráfico tiempo unitario búsqueda

La Figura 6.6 contiene el gráfico de los tiempos totales de búsqueda con diferente número de elementos. Se puede apreciar fácilmente que el crecimiento del tiempo no es proporcional al número de elementos, lo que significa lo mencionado anteriormente: cuantos más elementos se busquen, menos tiempo se tarda en encontrar un único elemento. El tiempo total contemplado en esta gráfica incluye la impresión por pantalla de los resultados, lo que incrementa en gran medida el tiempo en la búsqueda de un número de elementos elevado.

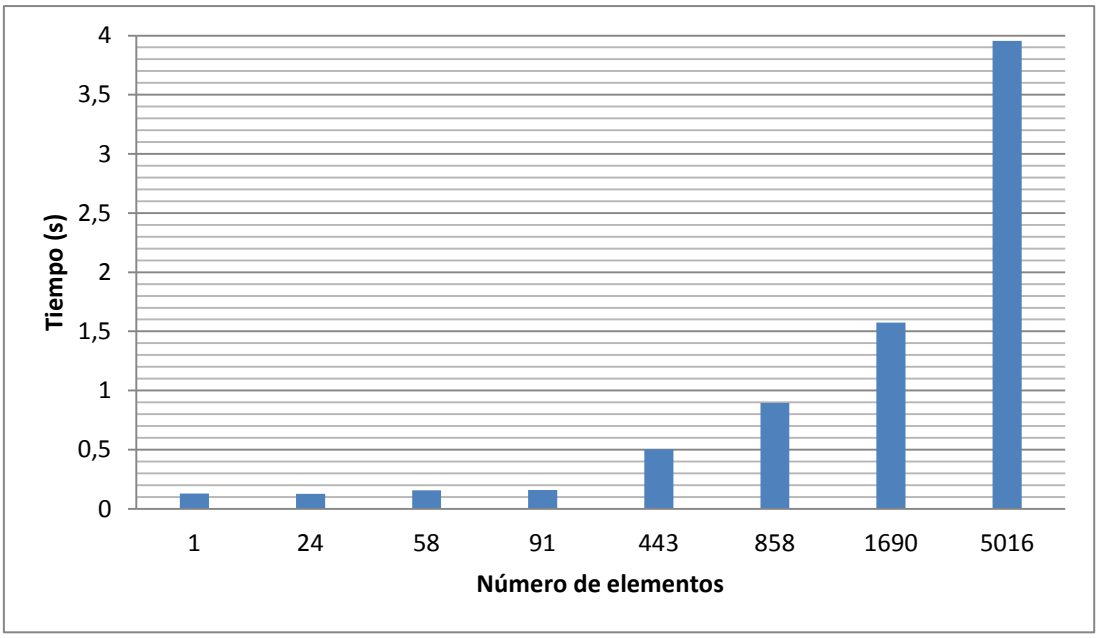

Figura 6.6: Gráfico tiempo total búsqueda

Los resultados de las pruebas demuestran que el sistema tiene un buen rendimiento en la operación buscar, lo que es importante ya que uno de los objetivos buscados era realizar un sistema que permitiera la localización rápida de los datos.

6.4 Conclusión general del análisis

La operación de inserción no tiene buen rendimiento debido al coste de calcular el md5 de cada archivo insertado en el sistema. Para solucionar este problema sería necesario encontrar otro método para identificar de forma unívoca los ficheros cuyo coste temporal sea menor. Una posible solución sería calcular el md5 de sólo una parte del fichero, no siendo necesaria la lectura del fichero completo, reduciendo así el tiempo de cálculo. Sin embargo de esta forma es más probable que haya colisiones, y por lo tanto no es adecuado para identificar un fichero de forma unívoca.

Por otra parte, las operaciones de búsqueda y eliminación ofrecen buen rendimiento. Se ha comprobado que el tiempo unitario con diferente número de elementos, es siempre el mismo en el caso de la operación eliminar y cada vez más bajo en la operación buscar.

Dado que el sistema tiene como objetivo facilitar la organización y localización de los archivos, y el número de inserciones no es elevado (un archivo se inserta una vez, y se localiza tantas como sea necesario), se puede concluir con que el rendimiento de dicho sistema es el adecuado.

7. Conclusiones

Se exponen en este apartado las conclusiones obtenidas en relación con los objetivos propuestos para este trabajo, las mejoras que pueden acometerse una vez alcanzados dichos objetivos básicos y un esbozo del esfuerzo realizado hasta llevar a término el proyecto.

7.1 Conclusiones generales

Tal y como se expone en la sección 1.2 Objetivos, el principal objetivo del presente trabajo era el análisis y diseño de un sistema de ficheros semántico, que facilite la organización del espacio de nombres y localización de los archivos del sistema, permitiendo al usuario el acceso transparente al sistema de ficheros. A lo largo de este documento se ha mostrado dicho análisis y diseño, cumpliendo por tanto el objetivo principal.

Así mismo, se han alcanzado los sub-objetivos propuestos:

- Se ha realizado un análisis de las tecnologías para gestionar los metadatos del sistema de ficheros, analizando sistemas de bases de datos *SQL* y *NoSQL*. Tras el análisis se ha optado por utilizar un sistema de bases de datos *NoSQL*, en concreto *MongoDB*.
- Se ha diseñado un espacio de nombres propio del sistema de ficheros, que utiliza tanto etiquetas definidas por el sistema como etiquetas gestionadas por el usuario.
- Se ha realizado un estudio de integración del sistema de ficheros en el sistema operativo Linux mediante el uso de FUSE, y se ha integrado parte de la funcionalidad del sistema a través de dicha herramienta.
- Se ha realizado el desarrollo ágil de un prototipo para la evaluación del sistema de ficheros analizado y diseñado. Dicho prototipo cumple con la funcionalidad analizada en los casos de uso y requisitos. El sistema de ficheros desarrollado permite:
    - Gestión de Etiquetas
        - Crear
        - Eliminar
        - Añadir
        - Modificar
        - Listar
    - Gestión de archivos
        - Insertar
        - Eliminar
        - Buscar

- Se ha evaluado el prototipo desarrollado, demostrando el buen rendimiento de este en la localización de los datos, cumpliendo así el objetivo principal del trabajo.

Se puede concluir, por lo tanto, que se han alcanzado con éxito todos los objetivos propuestos en el inicio de presente Trabajo Fin de Grado.

7.2 Trabajos Futuros

Dado que el sistema desarrollado es un prototipo, existen numerosos puntos en los que se puede seguir trabajando. A continuación se exponen algunos de ellos.

Hacer más escalable la operación insertar

En la evaluación del rendimiento del sistema, se comprobó que la operación insertar no tiene buen rendimiento debido al coste del cálculo del md5. Tal y como se mencionó en dicha evaluación, la solución al problema es encontrar otro método para identificar de forma unívoca los ficheros insertados en el sistema, que suponga menos tiempo. Un posible trabajo futuro sería hacer un estudio de algoritmos como SHA, Tiger, Hash, etc. evaluando el rendimiento tanto de los algoritmos exclusivamente, como de los algoritmos junto con la operación insertar, para encontrar una solución al problema mencionado y mejorar así el rendimiento del sistema.

Estudio profundo de integración con FUSE

Como se ha mencionado anteriormente en el presente documento, sólo se ha llegado a integrar con FUSE las operaciones de sólo lectura. Un posible trabajo futuro sería la realización de un estudio en profundidad de la integración con FUSE de todo el sistema de ficheros.

Más niveles de etiquetas

El sistema de ficheros desarrollado tiene un límite en el número de niveles de etiquetas posibles para su simplicidad. Una mejora a realizar sería eliminar este límite permitiendo tantos niveles de etiquetas como el usuario desee.

Integración del Sistema de Ficheros en Entornos Cloud

El sistema de ficheros ha sido diseñado y desarrollado para su utilización en local. Una posible línea de trabajo es la integración del sistema en un entorno *cloud*, permitiendo de esta forma la integración de sistemas de ficheros de distintos usuarios. Para el desarrollo de esta posibilidad, sería necesario el desarrollo de una interfaz web para la gestión de los datos,

desarrollo de una política de seguridad y un estudio legal sobre la protección de datos de carácter personal.

7.3 Presupuesto y Planificación

En esta sección se presenta el presupuesto del coste del presente trabajo así como la planificación resultante.

El presupuesto del presente TFG se muestra en el Anexo VII. El salario de los distintos roles se ha obtenido de [27].

En cuanto a la planificación final del proyecto, se contempla en el Anexo IX. Se puede observar que todas las tareas han sido completadas al 100%, acabando con éxito el trabajo.

8. Bibliografía

[1] *A tag-based filesystem for ubuntu*. (https://blueprints.launchpad.net/ubuntu/+spec/tag-based-filesystem) Abril 2012.

[2] Faubel, S. y Kuschel, C (2008).*Towards Semantic File System Interfaces*

[3] Prashanth Mohan, Raghuraman, Venkateswaran S y Dr. Arul Siromoney. *Semantic File Retrieval in File Systems using Virtual Directories*

[4] *Cómo se almacenan y organizan los archivos en Ubuntu 10.10* (http://www.lasticenelaula.es/portal/index.php/ubuntu10/la-gestion-de-archivos10/232-como-se-almacenan-y-organizan-los-archivos-en-ubuntu-1010.html) Mayo 2012.

[5] *Introducción a los sistemas de archivos FAT, HPFS y NTFS* (http://support.microsoft.com/kb/100108/es) Abril 2012.

[6] Jesús Carretero, Félix García, Pedro de Miguel y Fernando Pérez, 2001. *Sistemas Operativos: una visión aplicada*. (1ª edición). Mc Graw Hill

[7] *Design and Implementation of the Second Extended Filesystem* (http://e2fsprogs.sourceforge.net/ext2intro.html) Abril 2012.

[8] *Semantic File Systems.* (http://www.objs.com/survey/OFSExt.htm) Abril 2012.

[9] *Semantic Web Wiki* (http://semanticweb.org/wiki/SemFS) Abril 2012.

[10] *TAGXFS* (http://tagxfs.sourceforge.net/) Abril 2012.

[11] *WinFS 101: Introducing the New Windows File System* (http://msdn.microsoft.com/en-US/library/aa480687.aspx) Mayo 2012.

[12] *WinFSArch* (http://en.wikipedia.org/wiki/File:WinFSArch.svg) Mayo 2012.

[13] *Bases de Datos* (2003). En *Gran Temática PLANETA* (vol. 7, pp. 280-283). Editorial Planeta, S.A.

[14] *Bases de Datos* (http://blog.zwitchdesign.com/articulos/bases-de-datos/) Abril 2012.

[15] *Cuál es la diferencia entre una BD y un SGBD* (http://mayraalfonso.wordpress.com/2010/04/23/cual-es-la-diferencia-entre-una-bd-y-un-sgbd-sistema-de-gestion-de-base-de-datos/) Abril 2012.

[16] César Pérez, 2007. *MySQL para Windows y Linux* (2ª edición). Ra-Ma Editorial.

[17] *MongoDB* (http://www.mongodb.org/) Abril 2012.

[18] *Filesystem in Userspace* (http://fuse.sourceforge.net/) Abril 2012.

[19] Raúl González Duque. *Python para todos*. http://mundogeek.net/tutorial-python/

[20] José H. Canós, Patricio Letelier y Mª Carmen Penadés (2003). *Metodologías Ágiles en el Desarrollo de Software*. Actas de las VIII Jornadas de Ingeniería del Software y Sases de Datos, JISBD (pp. 1-8)

[21] *MESSY, EXCITING, AND ANXIETY-RIDDEN: ADAPTIVE SOFTWARE DEVELOPMENT* (http://www.adaptivesd.com/articles/messy.htm) Mayo 2012.

[22]*MongoDB vs SQL Server 2008 Duelo de rendimiento* (http://www.christianmania.com/2010/12/mongodb-vs-sql-server-2008-duelo-de-rendimiento/) Abril 2012.

[23] *optparse – Parser for command line options* (http://docs.python.org/library/optparse.html) Mayo 2012.

[24] *mongoDB. Python Language Center* (http://www.mongodb.org/display/DOCS/Python+Language+Center) Mayo 2012.

[25] *python 3* (http://pypi.python.org/pypi/pymongo/) Mayo 2012.

[26] *gnome-open: Open Anything from the Command Line* (http://embraceubuntu.com/2006/12/16/gnome-open-open-anything-from-the-command-line/) Mayo 2012.

[27] *Salario medio de un programador* (http://www.tufuncion.com/trabajo-programador) Mayo 2012.

Anexos

Anexo I: Manual de Instalación de CollectionFS

El sistema de ficheros CollectionFS ha sido desarrollado para su utilización en el Sistema Operativo Ubuntu. Para su correcto funcionamiento es necesaria la instalación previa de los siguientes componentes:

- Lenguaje de programación Python (Anexo II)
- Base de Datos *MongoDB* (Anexo III)
- Driver *PyMongo* (Anexo V)
- FUSE (Anexo IV)
- Gnome-open (Anexo V)

Una vez se han instalado los componentes mencionados y funcionan correctamente se puede proceder a la instalación de CollectionFS. Para ello es necesario abrir un Terminal y seguir los siguientes pasos.

1. Crear una carpeta que servirá como directorio raíz de todo el sistema de ficheros. Para ello es necesario colocarse en el directorio en el que se quiera crear (con el comando `cd`) y ejecutar la siguiente línea:

   ```
   $ mkdir <nombre directorio>
   ```

   Por ejemplo:

   ```
   $ mkdir CFS
   ```

   De esta forma se crea un directorio llamado CFS.

2. Copiar en el directorio creado los siguientes scripts (localizados en el directorio *src* del CD):

   a. CFSchange.py
   b. CFScopy.py
   c. CFSfind.py
   d. CFSinit.py
   e. CFSinsert.py
   f. CFSls.py
   g. CFSmkdir.py
   h. CFSmove.py
   i. CFSopen.py
   j. CFSprog.py
   k. CFSremove.py
   l. CFSrmdir.py
   m. CFSrmTag.py
   n. CollectionFS.py

3. Crear, dentro del directorio creado, dos nuevos directorios:

```
$ cd <directorio creado>
$ mkdir <nuevo directorio 1>
$ mkdir <nuevo directorio 2>
```

Por ejemplo, llamaremos a los nuevos directorios *Files* y *FS*.

```
$ cd CFS
$ mkdir Files
$ mkdir FS
```

El directorio Files es el componente de almacenamiento, donde se guardarán los archivos físicos insertados en el sistema. El directorio FS es sobre el que se montará el sistema de ficheros.

4. Inicializar el sistema, para ello ejecutar la siguiente línea:

```
$ python CFSinit.py /home/.../CFS/Files
```

Es necesario sustituir '**/home/.../**' por la ruta completa del directorio CFS.

5. Montar el sistema de ficheros:

```
$ python CollectionFS.py FS/
```

Una vez realizados los pasos anteriores, ya podrá empezar a utilizar CollectionFS (Anexo VI: Manual de Usuario). Si alguno de los pasos no se ha realizado correctamente, es necesario revisar si están instalados los componentes mencionados anteriormente, con la versión indicada en los respectivos anexos, y comenzar de nuevo con la instalación de CollectionFS.

Anexo II: Manual de Instalación de Python

Desde hace tiempo el lenguaje Python se encuentra pre-instalado en todas las distribuciones GNU/Linux. No obstante a continuación se explica cómo instalarlo.

Para la instalación es necesario abrir un terminal y escribir:

```
$ sudo apt-get install python
```

Después de unos minutos, Python ya estará listo para comenzar a utilizarlo.

Para conocer qué versión de Python ha sido instalada es necesario escribir en un terminal:

```
$ python -V
Python 2.7.2
```

Si la versión instalada es inferior a la versión 2.7.2 será necesario actualizarla.

Anexo III: Manual de Instalación de MongoDB

Para la instalación de *MongoDB* es necesario abrir un terminal y ejecutar la siguiente línea:

```
$ sudo gedit /etc/apt/sources.list
```

Se abrirá un fichero en el que se deberá añadir, al final, la siguiente línea:

```
deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen
```

Una vez añadida la línea anterior y guardado el fichero, es necesario ejecutar, en el terminal, las siguientes líneas:

- Añadir la clave GPF

```
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
```

- Actualizar la lista de paquetes

```
$ sudo apt-get update
```

- Instalar el paquete de MongoDB

```
$ sudo apt-get install mongodb-10gen
```

- Comprobar que se ha instalado correctamente introduciendo:

```
$ mongo
```

Si se ha instalado correctamente deberá aparecer lo siguiente:

```
MongoDB shell version: 2.0.5
    connecting to: test
    >
```

Nota: Es necesaria la instalación de una versión 2.0.x.

Anexo IV: Manual de Instalación de FUSE

Para la instalación de FUSE es necesaria la ejecución de la siguiente línea en un terminal:

```
$ sudo apt-get install fuse
```

Una vez instalado FUSE es necesario instalar el componente de python para FUSE:

```
$ sudo apt-get install python2.7-fuse
```

Anexo V: Manual de Instalación de Componentes

Para el correcto funcionamiento del sistema es necesario instalar los siguientes componentes:

- Gnome-open: componente que abre los archivos con los programas por defecto
- *PyMongo*: driver de Python para poder utilizar *MongoDB*.

Gnome-open

Para instalar este componente es necesario ejecutar la siguiente línea desde un terminal:

```
$ sudo apt-get install libgnome2-0
```

PyMongo

La instalación del driver P*yMongo* se realiza ejecutando en un terminal la siguiente línea:

```
$ sudo apt-get install python-pip
```

Una vez instalado el comando pip, se ejecuta lo siguiente para la instalación de PyMongo:

```
$ sudo pip install pymongo
```

Nota: para la correcta instalación del paquete *PyMongo* es necesario tener instalado GCC. Para su instalación ejecutar:

```
$ sudo apt-get install build-essential python-dev
```

Anexo VI: Manual de Usuario de CollectionFS

En este manual se proporcionan las instrucciones para el uso del Sistema de Ficheros CollectionFS.

Con CollectionFS usted puede organizar y localizar su información de una forma sencilla y rápida. Para ello el sistema permite la gestión de etiquetas y la gestión de archivos. Una vez instalado el sistema (Anexo I), es posible comenzar a utilizarlo. A continuación se proporciona una guía de las diferentes funcionalidades del sistema. Si tras la lectura de dicha guía aún le queda alguna duda, puede utilizar la ayuda de cada aplicación escribiendo el *flag* –h después del nombre de la aplicación.

Nota: para poder utilizar las aplicaciones se deberá situar en el directorio CFS en el que realizó la instalación.

Crear Etiquetas

Para crear una etiqueta es necesario utilizar la aplicación *CFSmkdir.py*. Será necesario introducir, por línea de comandos la etiqueta (ya creada) en la que se desea crear la nueva etiqueta y el nombre que se quiere dar a la etiqueta a crear.

```
$ python CFSmkdir.py <etiqueta padre> <nueva etiqueta>
```

Por ejemplo:

```
$ python CFSmkdir.py ETQ Libros
```

De esta forma, creamos una nueva etiqueta llamada *Libros* dentro de la etiqueta por defecto *ETQ* (recuerde que solo está permitida la creación de etiquetas dentro de la etiqueta por defecto *ETQ* o de sus sub-etiquetas).

Eliminar Etiquetas

Para eliminar una etiqueta del sistema es necesaria la utilización de la aplicación *CFSrmdir.py* de la siguiente forma:

```
$ python CFSrmdir.py <etiqueta>
```

Sustituyendo <etiqueta> por la etiqueta a eliminar. Si la etiqueta que desea eliminar contiene archivos y/o sub-etiquetas la aplicación le preguntará si desea eliminar la etiqueta y todo su

contenido. Si no permite la eliminación no se borrará ninguna etiqueta. Para eliminar la etiqueta y todo su contenido sin que la aplicación le pregunte puede añadir la opción –y.

```
$ python CFSrmdir.py <etiqueta> -y
```

Insertar Archivos

Si desea la inserción de archivos en el sistema es necesario que utilice la aplicación *CFSinsert.py*. Para ello deberá ejecutar en un terminal:

```
$ python CFSinsert.py <ruta del archivo>
```

Por ejemplo:

```
$ python CFSinsert.py /home/.../Archivo.pdf
```

Siendo */home/.../Archivo.pdf* la ruta (absoluta o relativa) del archivo que desea insertar. Una vez insertado el archivo en el sistema, se le asignarán automáticamente las etiquetas de extensión y tamaño.

Añadir Etiqueta

Para añadir una etiqueta (de las creadas por el usuario) a un archivo se utiliza la aplicación *CFScopy.py*. Es necesario introducir el ID del archivo al que desea añadir la etiqueta y la etiqueta a añadir.

```
$ python CFScopy.py <id> <etiqueta>
```

Siendo <id> el ID del archivo y <etiqueta> la etiqueta a añadir. Para saber el ID de un archivo puede utilizar la aplicación *CFSls.py* (lista todos los archivos del sistema) o puede realizar una búsqueda del archivo con la aplicación *CFSfind.py*.

Cambiar Etiqueta

Si desea cambiar una etiqueta de un archivo por otra, podrá utilizar la aplicación *CFSmove.py*. Es necesario introducir el ID del archivo en el que quiere cambiar las etiquetas, la etiqueta a cambiar y la etiqueta por la que desea cambiarla; de la siguiente forma:

```
$ python CFSmove.py <id> –f <etiqueta origen> –t <etiqueta destino>
```

```
$ python CFSmove.py 4fc630f51d41c81403000000 –f Libros –t Artículos
```

De esta forma se cambiará la etiqueta *Libros* por la etiqueta *Artículos* en el archivo con ID *4fc630f51d41c81403000000*. El orden de ambas etiquetas puede ser el que desee siempre que sigan al *flag* correspondiente:

- -f : Etiqueta a cambiar
- -t: Etiqueta por la que se desea cambiar

Eliminar Etiqueta de Archivo

Para eliminar una etiqueta de un archivo en concreto, es necesario utilizar la aplicación *CFSrmTag.py*. Dicha aplicación tiene dos posibilidades:

- Eliminar todas las etiquetas de un archivo (excepto las etiquetas por defecto extensión y tamaño)

```
$ python CFSrmTag.py <id> -a
```

- Eliminar una etiqueta en concreto de un archivo

```
$ python CFSrmTag.py <id> -t <etiqueta>
```

Listar Elementos del Sistema

Para listar los elementos que contiene el sistema, tanto etiquetas como archivos se utiliza la aplicación *CFSls.py*. Con esta aplicación se pueden realizar varios tipos de listados:

- Listar todas las etiquetas y archivos del sistema:

```
$ python CFSls.py -a
```

El resultado de ejecutar la línea anterior es parecido al siguiente:

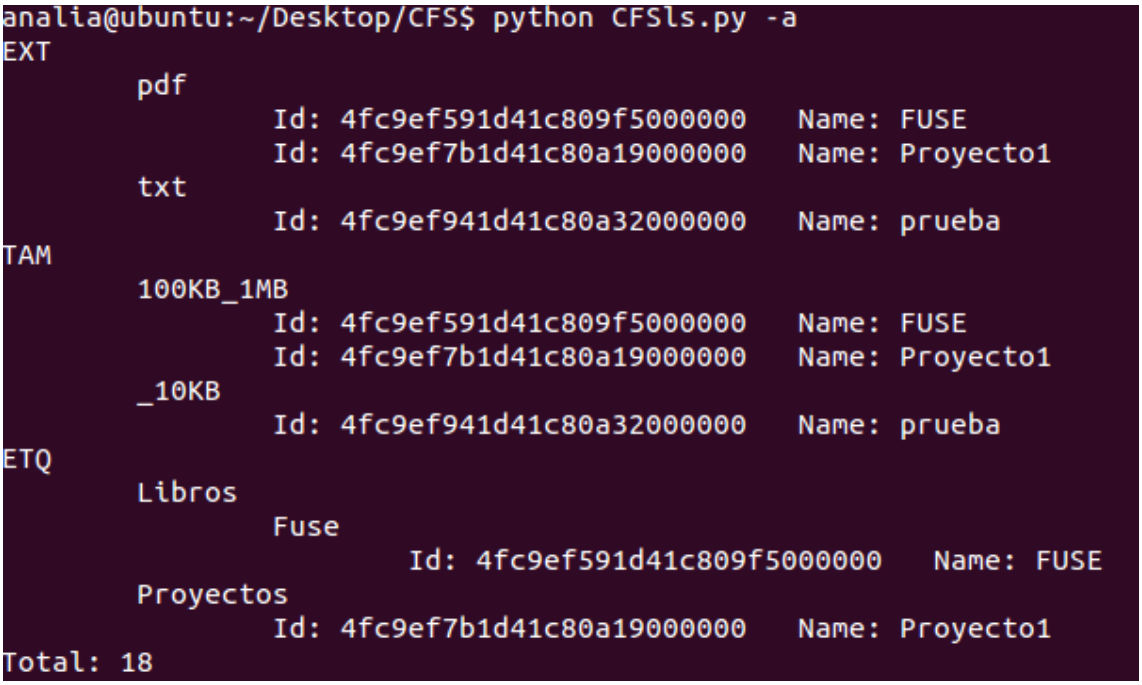

Figura Anexos.1: Listar contenido del sistema

- Listar todas las etiquetas del sistema:

```
$ python CFSls.py -e
```

- Listar todos los archivos del sistema:

```
$ python CFSls.py -f
```

- Listar los archivos y etiquetas que contiene una etiqueta

```
$ python CFSls.py <etiqueta> -a
```

De esta forma se listarán todos los archivos y etiquetas por debajo de <etiqueta>.

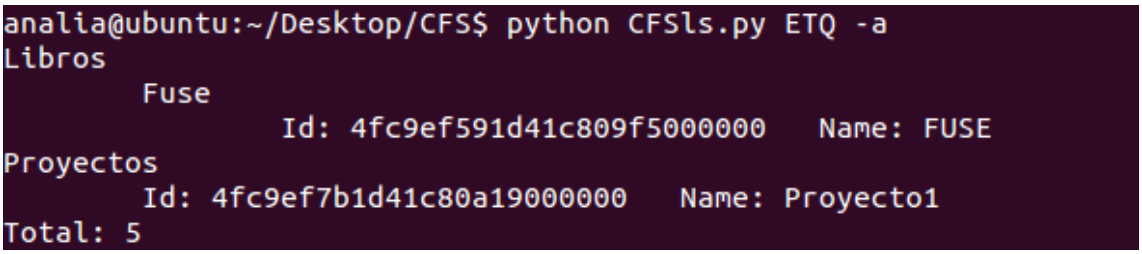

Figura Anexos.2: Listar contenido etiqueta

- Listar las etiquetas que contiene una etiqueta.

```
$ python CFSls.py <etiqueta> -e
```

- Listar todos los archivos que contiene una etiqueta.

```
$ python CFSls.py <etiqueta> -f
```

Buscar Archivos

El Sistema de Ficheros CollectionFS permite realizar búsquedas por múltiples campos. Para ello se utiliza la aplicación *CFSfind.py* con las siguientes opciones:

- -n: búsquedas por nombre.
- -e: búsquedas por extensión.
- -g: búsquedas por tamaño mayor que el introducido.
- -l: búsquedas por tamaño menor que el introducido.
- -t: búsquedas por etiquetas. Hasta tres etiquetas en la misma búsqueda.
- -m: búsquedas por md5.
- -d: búsquedas por descripción.

En la búsqueda por nombre y por descripción no es necesario introducir las palabras exactas, es posible hacer búsquedas por aproximación.

Para hacer búsquedas es necesario ejecutar la aplicación con los *flags* que se deseen buscar, siendo posible utilizar en cada búsqueda como mínimo un *flag* y como máximo todos. En la Figura Anexos.3 se muestra un ejemplo de búsqueda y sus resultados.

```
analia@ubuntu:~/Desktop/CFS$ python CFSfind.py -n fu -e pdf

Name: FUSE
Id: 4fc9ef591d41c809f5000000
Extension: pdf
Size: 100KB_1MB
Path: /home/analia/Desktop/CFS/Files/4fc9ef591d41c809f5000000.dat
MD5: 9eae4822f1214342116fdf728543535f
Tags: [{'Libros': 'Fuse'}]
Description:
------------------------------------------------
```

Figura Anexos.3: Ejemplo búsqueda

Cambiar Nombre y Añadir Descripción

CollectionFS permite cambiar el nombre a los archivos insertados, así como añadir una descripción. Para ello se utiliza la aplicación *CFSchange.py*. Para cambiar el nombre es necesario ejecutar la siguiente línea:

```
$ python CFSchange.py <id> –n <nombre>
```

Por ejemplo:

```
$ python CFSchange.py 4fc630f51d41c81403000000 –n Nombre
```

De esta forma se habrá cambiado el nombre del archivo *4fc630f51d41c81403000000* a *Nombre*. Para añadir una descripción a un archivo:

```
$ python CFSchange.py <id> –d
Insert the file description:
```

Añadiendo la descripción a continuación y pulsando 'Enter', se habrá añadido dicha descripción al archivo <id>.

Nota: es posible cambiar el nombre y añadir la descripción de un archivo con la misma llamada, no importando el orden de los *flags*.

```
$ python CFSchange.py <id> –n <nombre> –d
Insert the file description:
```

Eliminar Archivo

Para eliminar un archivo del sistema se utiliza la aplicación *CFSremove.py*. Funciona de la siguiente forma:

```
$ python CFSremove.py <id>
```

De esta forma se habrá eliminado el archivo con id <id> del sistema.

Abrir Archivo

El Sistema de Ficheros CollectionFS permite abrir los archivos insertados. Para ellos se utiliza la aplicación *CFSopen.py* de la siguiente forma:

```
$ python CFSopen.py <id>
```

Una vez ejecutada la línea anterior se abrirá el archivo con id <id> con el programa que el usuario ha elegido para abrir la extensión del archivo (*CFSprog.py*) o con el programa por defecto.

Configurar Programas

CollectionFS permite configurar los programas con los que se abrirán las diferentes extensiones de archivo. Para ello se utiliza la aplicación *CFSprog.py*:

```
$ python CFSprog.py
```

Un ejemplo de utilización:

```
$ python CFSprog.py
extension: pdf
program: Evince
Continue or Stop? (c/s)
```

De esta forma, quedará configurado en el sistema que para los archivos *pdf* el programa a utilizar para abrir es *Evince*. Si se continúa (introduciendo *c*) se seguirán introduciendo extensiones y programas hasta que el usuario desee parar (introduciendo *s*).

Anexo VII: Presupuesto

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1. Autor:
   Analía de Pedro Santamaría
2. Departamento:
   Informática
3. Descripción del Proyecto:
   - Título **CollectionFS: Sistema de Ficheros basado en Etiquetas**
   - Duración (meses) 3
   - Tasa de costes Indirectos: 20%
4. Presupuesto total del Proyecto (valores en Euros):
   8.142,00 Euros

5. Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar solo a título informativo)	Categoría	Dedicación (hombres mes) [a)]	Coste hombre mes	Coste (Euro)
de Pedro Santamaría, Analía		Analista	1	3.000,00	3.000,00
de Pedro Santamaría, Analía		Diseñador	1	2.200,00	2.200,00
de Pedro Santamaría, Analía		Programador	1	1.300,00	1.300,00
	Hombres mes 3			**Total**	6.500,00

[a)] 1 Hombre mes = 131.25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8.8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d)]
Ordenador Intel(R) Core™ i7	700,00	100	3	60	35,00
					0,00
				Total	35,00

[d)] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} xCxD$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C= coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO [e)]

Descripción	Empresa	Coste imputable
Internet	Movistar	60,00
Microsoft Office 2007	Microsoft	130,00
Luz	Endesa	60,00
	Total	250,00

[e)]Este capítulo de gastos incluye todos los gastos no contemplados en conceptos anteriores, por ejemplo: fungible, viajes y dietas

6. Resumen de costes

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	6.500
Amortización	35
Subcontratación de tareas	0
Costes de funcionamiento	250
Costes Indirectos	1.357
Total	8.142

Anexo VIII: Planificación Inicial

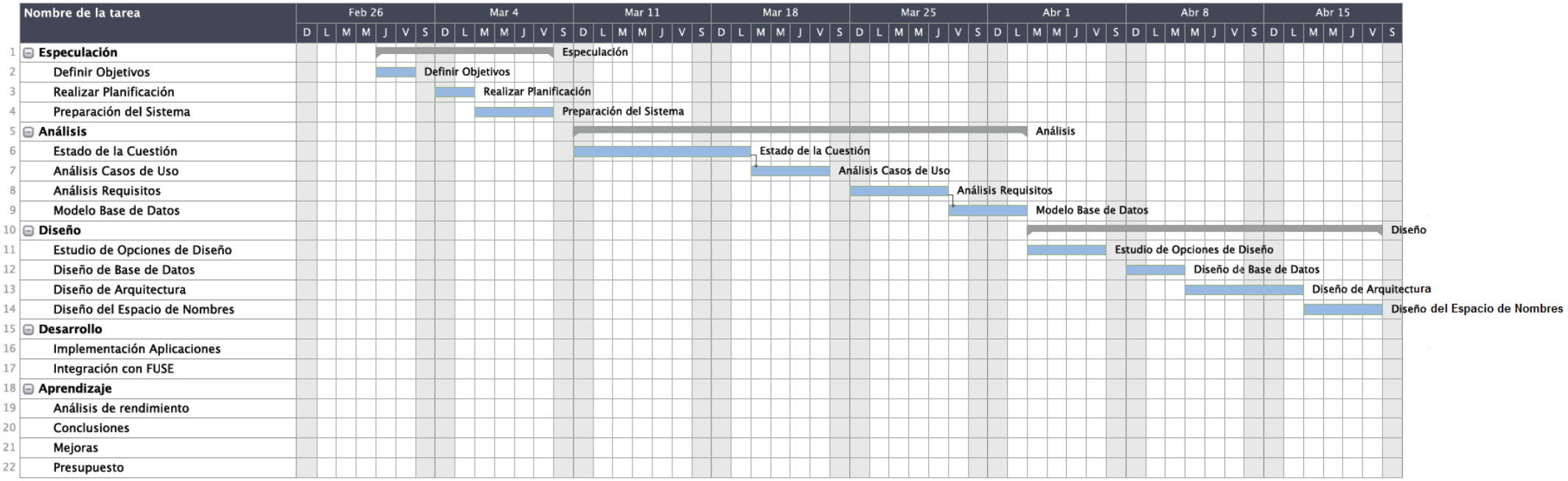

Figura Anexos.4: Planificación Inicial (Primera parte)

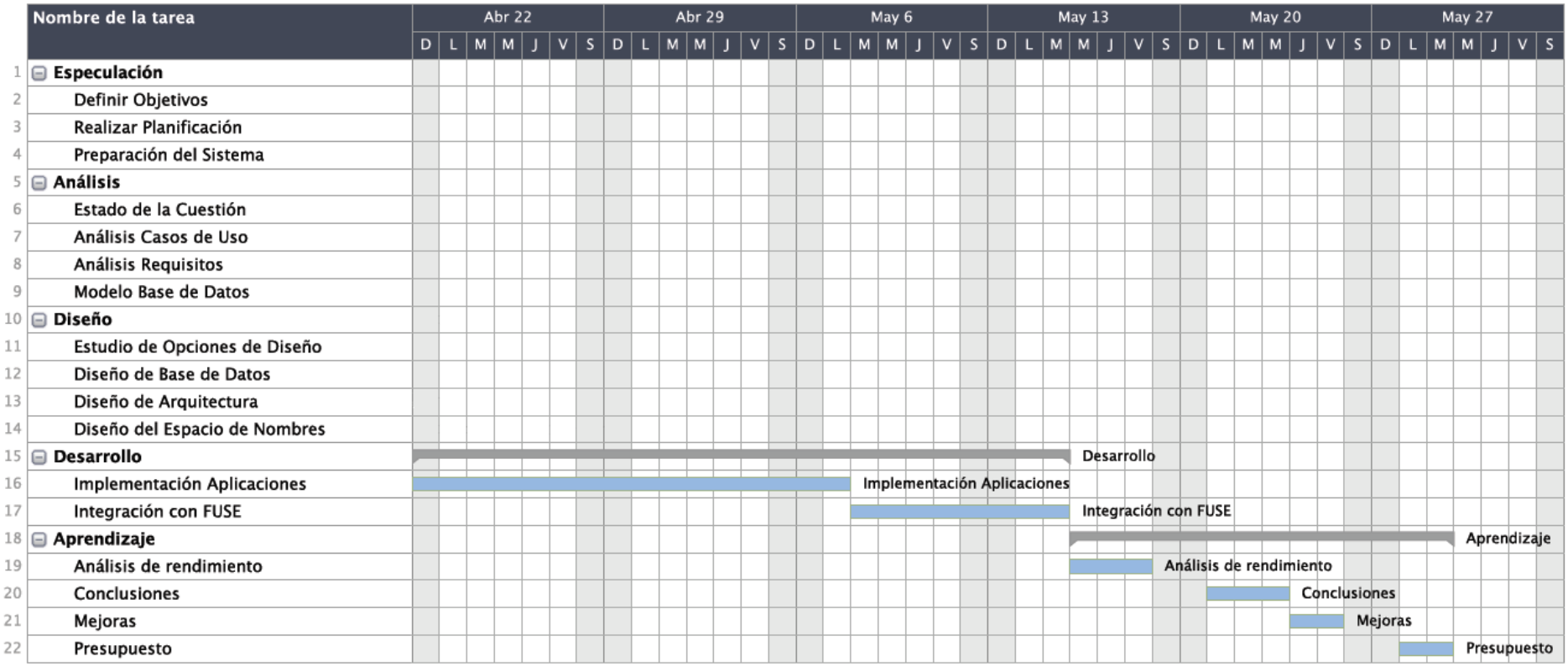

Figura Anexos.5: Planificación Inicial (Segunda parte)

Anexo IX: Planificación Final

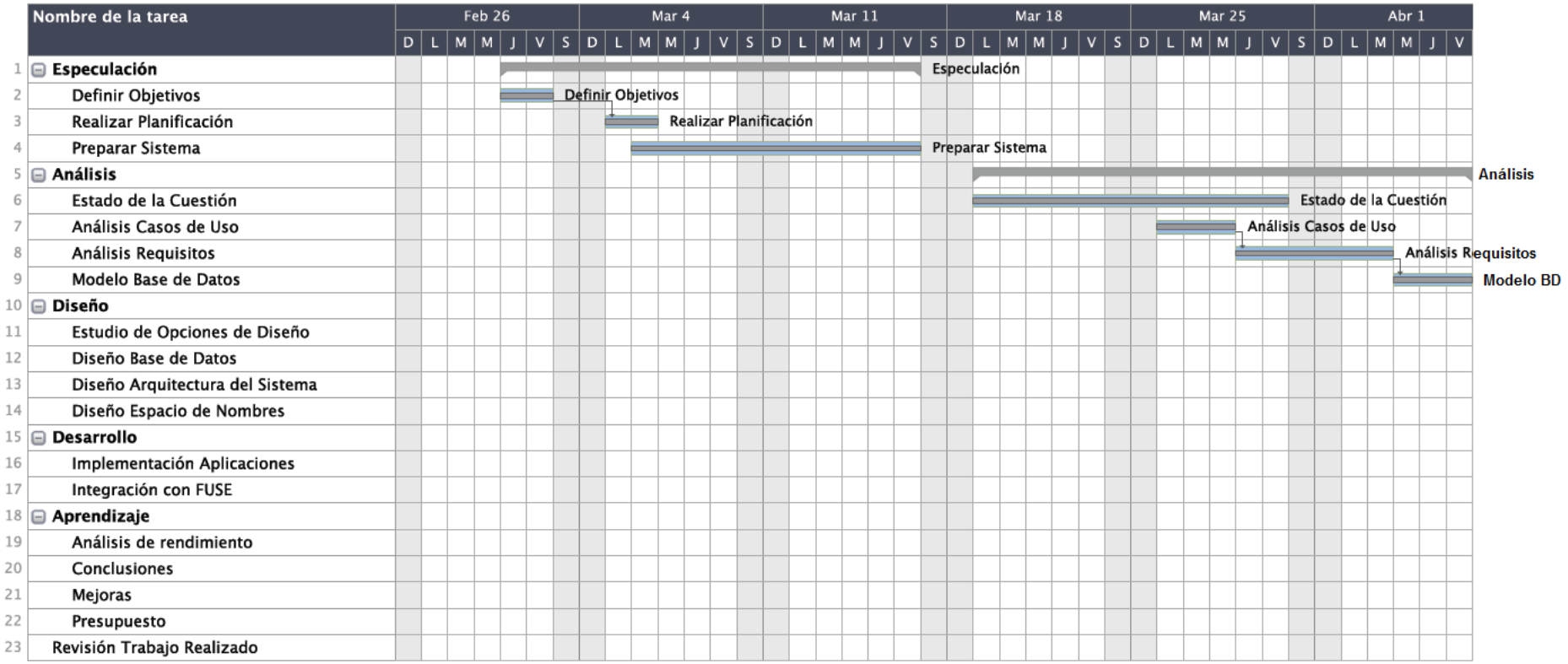

Figura Anexos.6: Planificación Final (Primera parte)

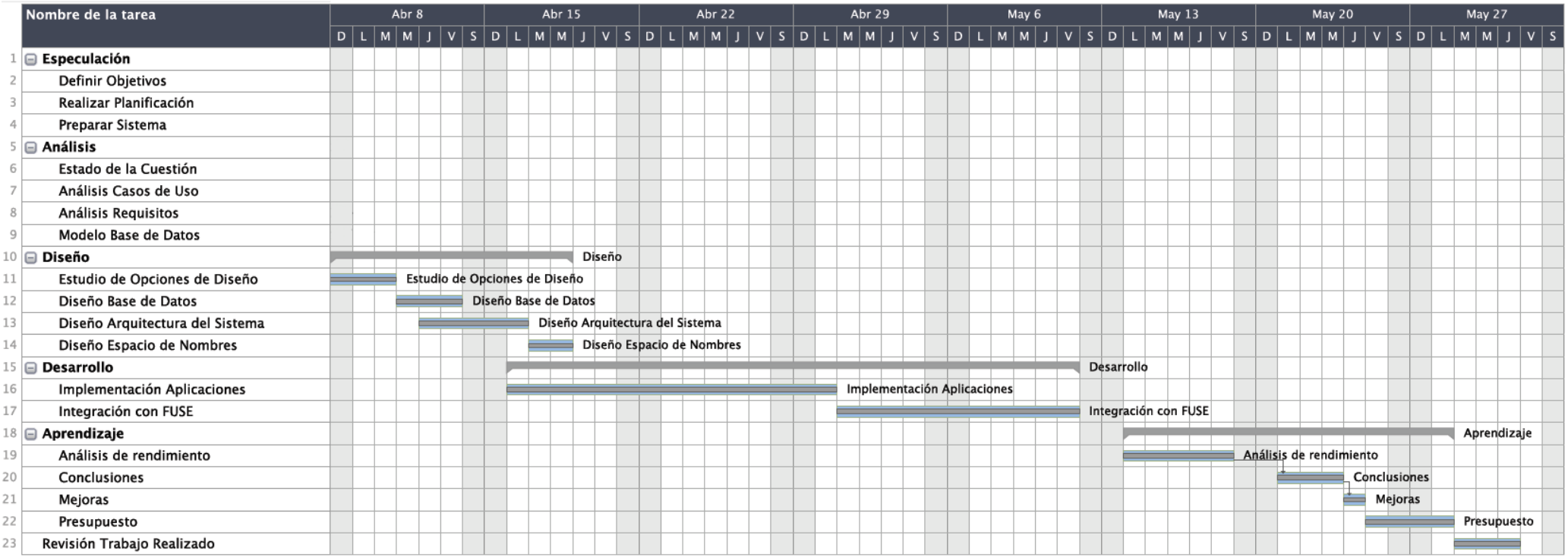

Figura Anexos.7: Planificación Final (Segunda parte)